ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20140519212813

SOFTWARE PROJECT SCHEDULING USING THE HYPER-CUBE ANT COLONY OPTIMIZATION ALGORITHM

Broderick Crawford, Ricardo Soto, Franklin Johnson, Sanjay Misra, Fernando Paredes, Eduardo Olguín

Original scientific paper
This paper introduces a proposal of design of Ant Colony Optimization algorithm paradigm using Hyper-Cube framework to solve the Software Project Scheduling Problem. This NP-hard problem consists in assigning tasks to employees in order to minimize the project duration and its overall cost. This assignment must satisfy the problem constraints and precedence between tasks. The approach presented here employs the Hyper-Cube framework in order to establish an explicitly multidimensional space to control the ant behaviour. This allows us to autonomously handle the exploration of the search space with the aim of reaching encouraging solutions.

Keywords: *Ant Colony Optimization; Hyper-Cube; scheduling; Software Project Management*

Programiranje računarskog projekta primjenom Hyper-Cube algoritma za optimizaciju kolonije mrava

Izvorni znanstveni članak
U radu se daje prijedlog dizajna paradigme algoritma za optimizaciju kolonije mrava primjenom Hyper-Cube sustava za rješenje problema programiranja računarskog projekta (Software Project Scheduling Problem). Taj se NP-hard problem sastoji od davanja zaduženja zaposlenicima u svrhu smanjenja trajanja projekta i njegovih ukupnih troškova. To zaduženje mora zadovoljiti ograničenja problema i pitanje prvenstva među zadacima. Pristup prikazan ovdje koristi Hyper-Cube sustav za uspostavljanje eksplicitno multidimenzionalnog prostora za kontrolu ponašanja mravi. Time nam se omogućava autonomno vođenje istraživanja u cilju pronalaženja ohrabrujućih rješenja.

Ključne riječi: *Hyper-Cube; optimizacija kolonije mrava; programiranje; vođenje računarskog projekta*

1 Introduction

Every software project has its manager and this project manager needs to apply his knowledge to solve multiple problems. This is a hard job for a single person and this consumes time excessively. Generally, the Software Project Scheduling Problem (SPSP) consists in finding a worker-task schedule, so that the resource constraint and task precedence are fulfilled. In addition the duration and cost of the project should be minimized [1].

This is a NP-hard problem, with multiple combinatorial optimization issues and needs much time to be completed. For this reason, it is attractive to create algorithms to automate this complex task.

In this paper we present a proposal of a design of an Ant Colony Optimization (ACO) algorithm. This proposal is complemented with Hyper-Cube (HC) framework and is used to solve the SPSP. This algorithm is called Max-Min Ant System with Hypercube framework (MMAS-HC). The SPSP is a variant of a well-documented problem called Resource-Constrained Project Scheduling (RCPS) [2]. SPSP has a set of tasks. These tasks must be performed in a specific order. A group of employees should be assigned to each task according to their personal skills. This should be done in order to minimize the project duration and its overall cost.

SPSP and RCPS have some differences. SPSP has one more objective, as minimize the cost associated to the employees, another difference is that employees have multiple skills, which are not quantifiable objects. The employees can have or not these skills. Whereas RCPS has multiple quantifiable resources. The employees work in a set of tasks, in order to meet the skills required by the tasks. Additionally they receive a monthly salary. For this reason SPSP is more realistic than RCPS. Some techniques used for this problem are based in metaheuristics and some priority rules [3, 4].

ACO is a well-known technique, based on the real ant colonies. In ACO, a group of software components called artificial ants travel in a graph to find good solutions to a given optimization problem. This bio-inspired metaheuristic is able to approximately solve several NP-hard combinatorial problems effectively [5 ÷ 8].

The Hyper-Cube framework was initially proposed by Blum et al. [9]. With this framework the limits of pheromone values are handled automatically. We implement this framework for the robustness of our algorithm and facilitating implementation [10, 11, 9]. This framework can be used on several ACO algorithms.

We believe that the constructive nature of ACO is effective to attacking SPSP. This idea is based on the fact that ACO can handle successfully a graph based search problem and SPSP is easily represented as this kind of problem. Another motivation is that ACO has demonstrated to solve successfully other scheduling problems such as University Course Timetabling Problem, Nurse Scheduling Problem, and Job Scheduling Problem among others [5, 10, 12, 13].

This paper is organized as follows. Section 2 presents the definition of the problem, section 3 presents a brief description of ACO, section 4 presents the design of an MMAS-HC for SPSP. In section 5 we present the preliminary results. The conclusions are presented in section 6.

2 The SPSP

The manager is faced with a variety of problems in a real software project. One of the most common is the assignment of employees to different tasks of the project. These tasks are established in the Gantt chart [11].

The Software Project Scheduling Problem should consider the employees, the tasks, and the skills [14, 15].The first model of the problem is defined in [16]. The tasks are the activities needed to end the project and require specific skills. The employees work in the tasks and they have some or all of these skills. The goal is to assign employees to tasks in order to minimize the project duration and its overall cost.

2.1 SPSP model

To understand the SPSP, we describe the model of the problem. First we describe the principal resources, and later the mathematical model.

The skills. The skills are the features that the employees have, for example: expert analyst, leadership, GUI designer, expert in some programming language, among others. To accomplish the tasks a set of skills is required. The group of all skills required in a project is defined as $Skl=\{sk_1,...,sk_{|Skl|}\}$, where $|Skl|$ is the maximun of skills.

The tasks. All jobs needed in the project are defined as a set of tasks, $Tsk=\{tsk_1,tsk_2,...tsk_{|Tsk|}\}$, where $|Tsk|$ is the number of tasks. For example: requirements analysis, system design, implementation, and deployment. The task j is composed by two attributes: tsk_j^{skl} is the set of skills necessary for activity j,$tsk_i^{skl} \subseteq Skl.tsk_j^{eff}$ represents the workload of the activity j.

These tasks have different precedence among them. We use an acyclic directed graph defined as PG (Precedence Graph) to represent their precedence [6, 7]. The PG is denoted as $G=(V, E)$, where V is the set of tasks, E is a set of edges and represent the precedence relation, an edge$(tsk_i,tsk_j)\in E$, that is tsk_i is a predecessor of tsk_j task.

The employees. The employees are the most important component in the problem. They have software engineering skills. The employees are assigned to tasks. They must also comply with the required skills for the task. $EM=\{emp_1,...,emp_{|Emp|}\}$ is the set of employees, where $|Emp|$ is the number of employees. An employee i is composed by three attributes: emp_i^{skl} is the set of employee skills, $emp_i^{skl} \subseteq Skl.emp_i^{maxd}$, is the maximum dedication to the project. $emp_i^{maxd} \in [0,1]$, when $emp_i^{maxd}=1,0$ the employee works his whole day in the project, if emp_i^{maxd} is less that 1, the employee works on the project partially. emp_i^{rem} is the salary of employee, and it is represented by a real number.

The skills, tasks, and employees are used to define the SPSP model. This model was originally proposed in [16] and then used in [14] with an ACS algorithm.The elements of the model are described in Tab. 1.

To represent a solution, a matrix $\boldsymbol{M}=(m)_{|Emp|\times|Tsk|}$ is used. The matrix $\boldsymbol{M}$ is determined by $|Emp|$ and $|Tsk|$. The elements of the matrix m_{ij} are real numbers, where $0 \le m_{ij} \le 1$,m_{ij} is the dedication of employee i to task j. When $m_{ij}= 0,0$, the employee i does not work on the task j, this indicates that the employee is available to work on another task. If $m_{ij} = 1,0$, the employee i is fully dedicated to the task j. For example, when $m_{ij} = 0,25$, the employee i employs 25 % of his time in the task j.

The solutions can be generated by different techniques, for example these can be constructed randomly or using a heuristics. After a solution is generated, this should fulfil some hard constraints to be feasible. These constraints are listed below.

- The first constraint must ensure that all tasks are completed, so that each task must be assigned to at least one employee. This means that the sum of each column must be greater than zero. This is presented in Eq. (1).

Table 1 Elements of the model

Element	Definition
$Skl=\{skl_1,...,skl_{|Skl|}\}$	Skills set required for the project
$Tsk=\{tsk_1, ...,tsk_{|Tsk|}\}$	Tasks set involved in the project
$G(V,E)$	Represent the PG
$V=\{tsk_1,...,tsk_{|Tsk|}\}$	Vertex set of the PG
$E=\{(tsk_i,tsk_j),...,(tsk_n,tsk_{|Tsk|})\}$	Edge set of the PG
tsk_j^{skl}	Skills set for the task j
tsk_j^{eff}	Monthly effort to perform the task j
$EM=\{emp_1, ...,emp_{|Emp|}\}$	Employees set working on the project
emp_i^{skl}	Skills set of the emp_i
emp_i^{maxd}	Maximum dedication to the project
emp_i^{rem}	Monthly salary of emp_i
$\boldsymbol{M}$	Matrix to represent a solution
tsk_j^{init}	The starting time of task j
tsk_j^{term}	The ending time for task j
tsk_j^{cos}	Calculated cost for the task j
tsk_j^{len}	Calculated duration for the task j
pjt^{len}	Whole project duration
pjt^{cos}	Overall cost of the project
emp_i^{overw}	Overburdening of employee emp_i
pjt^{overw}	Overburdening of project

$$\sum_{i=1}^{|Emp|} m_{ij} > 0 \qquad \forall j \in \{1,...,|\,Tsk\,|\}\,. \tag{1}$$

- The second constraint ensures that employees have the skills to perform the assigned tasks.The skills needed to resolve a task tsk_j^{skl} are a subset of Skl. Then each employee assigned to the task j must have one or all of the abilities required by the task. To accomplish that, tsk_j^{skl} must be a subgroup of the union of skills of employees who work on the task j. This means that, for all $m_{ij} > 0$, tsk_j^{skl} is a subset of the union emp_i^{skl}, with $i \in \{1,..,|Emp|\}$ for $\forall j \in \{1,..,|Tsk|\}$.

To better understand this model, we describe an example which consists of a set of tasks $TSK=\{tsk_1,tsk_2,tsk_3,tsk_4,tsk_5\}$, with task number $|tsk|=5$, a set of employees $EM=\{emp_1,emp_2,emp_3, emp_4\}$, with employee number $|emp| = 4$. The PG of the project is presented in Fig. 1. Further, the skills tsk^{skl} needed for each task and the effort tsk^{eff} required to complete them are presented.

$emp_1^{skl}=\{skl_1,skl_3\}$ $\quad$ $emp_2^{skl}=\{skl_1,skl_2,skl_3,skl_4\}$
$emp_1^{maxd}=1$ $\quad$ $emp_2^{maxd}=1$
$emp_1^{rem}=\$10$ $\quad$ $emp_2^{rem}=\$20$

$emp_3^{skl}=\{skl_1,skl_4\}$ $\quad$ $emp_4^{skl}=\{skl_3\}$
$emp_3^{maxd}=1$ $\quad$ $emp_4^{maxd}=1$
$emp_3^{rem}=\$10$ $\quad$ $emp_4^{rem}=\$10$

Figure 2 Properties of employees for the example in Fig. 1

In Fig. 2, we present the properties of employees, with their skills emp^{skl}, monthly salary emp^{rem}, and maximum dedication emp^{maxd}.

In Fig. 3, we present a possible solution for the example. The value of m_{ij} of matrix $\boldsymbol{M}$ represents the possible value for the dedication of an employee i to the task j. Each task j is assigned to at least one employee i and the employees assigned to a task have all necessary skills to carry out the job.

After obtaining the matrix $\boldsymbol{M}$, we are able to check whether the solution is feasible. Then we have to compute the duration of the whole project and its cost. With this information we can compare the quality of the solution.

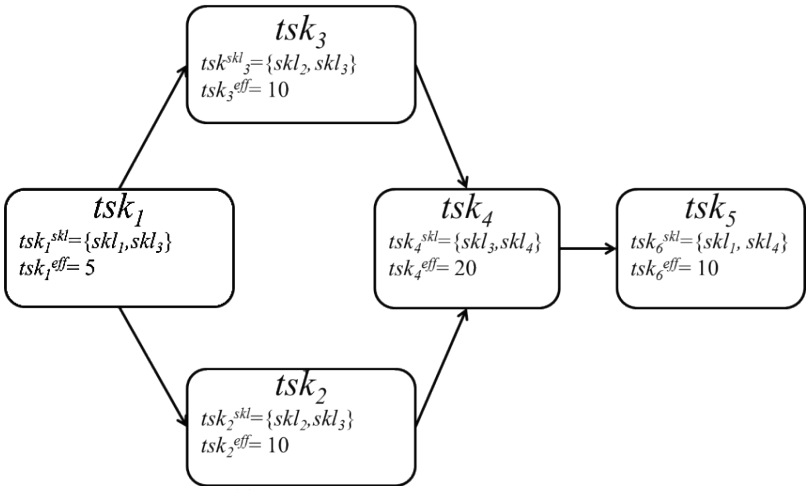

Figure 1 PG with $Tsk=\{tsk_1,tsk_2,tsk_3,tsk_4,tsk_5\}$, $Skl=\{skl_1,skl_2,skl_3,skl_4\}$

To compute the duration of the whole project, we have to obtain the duration of each task $tsk_j^{len}, j \in \{1,...,|Tsk|\}$. For this we use the monthly effort tsk_j^{eff} and the sum of the dedication of each employee assigned to the task j. This is presented in Eq. (2).

	tsk_1	tsk_2	tsk_3	tsk_4	tsk_5
emp_1	1,00	0,00	1,00	0,00	0,25
emp_2	0,00	1,00	0,00	0,25	0,00
emp_3	0,00	0,00	0,00	0,00	1,00
emp_4	0,00	0,50	0,50	1,00	0,00

Figure 3 Example of a solution represented in matrix $\boldsymbol{M}$

$$tsk_j^{len} = \frac{tsk_j^{eff}}{\sum_{i=1}^{|Emp|} m_{ij}}. \quad (2)$$

With the duration of the task and the precedence information in PG, we can obtain the starting time tsk_j^{init} and the ending time tsk_j^{term} for task j. We must consider two cases: first, when the task has not precedence, the staring time is tsk_j^{init} = 0,0. Second, when the task has precedence, the starting time is equal to the ending time tsk_k^{term}, where tsk_k^{term} is the ending time of the most belated preceding task. tsk_j^{init} is defined as follow:

$$tsk_j^{init} = \begin{cases} 0 & if\ \forall k \neq j, (tsk_k, tsk_j) \notin E \\ \max\left\{ tsk_k^{term} \middle| (tsk_k, tsk_j) \in E \right\} & \text{else} \end{cases}. \quad (3)$$

To obtain the ending time tsk_j^{term} for a task j, we need sum the starting time tsk_j^{init} and the duration of the task tsk_j^{len}, such as shown in Eq. (4).

$$tsk_j^{term} = tsk_j^{init} + tsk_j^{len}. \quad (4)$$

To calculate the total duration of the project, we use the Gantt chart information and the duration of each task in the project. For this we just need the ending time of last task. We can calculate it as follows:

$$pjt^{len} = \max\left\{ tsk_l^{term} \middle| \forall l \neq j\ (t_j, t_l) \notin E \right\}. \quad (5)$$

When we calculate the overall cost of the project pjt^{cos}, the cost of each task of the project must be previously obtained. The cost tsk_j^{cos} for the task j can be calculated using the Eq. (6), and then the overall cost of the project pjt^{cos} is obtained by summing each of the costs of the tasks using the Eq. (7).

$$tsk_j^{\cos} = \sum_{i=1}^{|Emp|} emp_i^{rem} \cdot m_{ij} \cdot tsk_j^{dur}, \quad (6)$$

$$pjt^{\cos} = \sum_{j=1}^{|Tsk|} tsk_j^{\cos}. \quad (7)$$

To meet the goal of the problem, we must define an objective function. The objective function is based on the minimization of the fitness of the solution. The fitness is associated with the duration of the whole project pjt^{len} and its overall cost pjt^{cos}. Then, the objective function is given in Eq. (8), where wp^{cos} and wp^{len} represent the importance of pjt^{cos} and pjt^{len} respectively. wp^{cos} and wp^{len} are parameters.

$$Min\ f^{fit}(x) = (wp^{\cos} pjt^{\cos} + wp^{len} pjt^{len}). \quad (8)$$

The problem presents an additional constraint, associated with the overburdening of an employee. This may increase the overall cost and consequently increase the fitness function. We define the work of an employee i as emp_i^w. To obtain emp_i^w, we use the matrix $\boldsymbol{M}$ and the

starting and ending time of the tasks assigned to the employee i at time t. The function is defined as follows.

$$emp_i^w = \sum_{\left\{j \middle| tsk_j^{init} \le t \le tsk_j^{term}\right\}} m_{ij} \quad . \tag{9}$$

The work of the employee i at instant t defined as emp_i^w, must not exceed the maximun deddication of the employee i to the project. That means, $emp_i^w(t) < emp_i^{maxd}$. To obtain the overburdening of employee i, we define a function as follows:

$$r(x) = \begin{cases} x & if\ x > 0 \\ 0 & if\ x \le 0 \end{cases}. \tag{10}$$

Then, using the function $r(x)$, we can determine the behaviour when exists overburdening or not. Finally, we use the Eq. (10) in Eq. (11) to obtain the overburdening for the employee i. That is determined as emp_i^{overw}.

$$emp_i^{overw} = \int_{t=0}^{t=pjt^{len}} r(emp_i^w(t) - emp_i^{maxd})\mathrm{d}t. \tag{11}$$

A solution is feasible if it satisfies all the previously defined constraint and also the overburdening of the project is equal to 0, that means $pjt^{overw} = 0$. To calculate project the overburdening pjt^{overw}, we must add the overburdening of each employee involved in the project. We can use the following formula:

$$pjt^{overw} = \sum_{i=1}^{|Emp|} emp_i^{overw} \quad . \tag{12}$$

3 Ant Colony Optimization metaheuristic

The ant colony optimization metaheuristic is a swarm intelligence technique inspired by the collection of food by natural ant colonies. This is an efficient probabilistic technique capable of solving different problems that can be mapped as a path through a graph.

Ant colony optimization is an algorithm based on a graph representation in which an artificial colony of ants collaborates to find solutions to discrete combinatorial problems [7, 17, 25]. The collaboration is a principal design component of ACO algorithms: A set of computational ants communicate using pheromones, this means using an indirect mechanism to guide its travel [18]. The ants cross the construction graph from a starting point to the ending point. The ants choose the nodes during their trip according to a probabilistic function. The probabilistic function is given by heuristic information of the problems, the pheromone information, and specific parameters such as alpha and beta [19, 20]. This algorithm was used successfully to solve various difficult combinatorial problems in optimization [21].

3.1 HC framework

Initially, the HC framework was presented by Blum et al. [9]. The HC framework can be applied to different Ant Colony Optimization algorithms, in which the possible values of pheromone are delimited to a multidimensional space.

The multidimensional space is defined as a convex hull where the feasible solutions are coded.

HC changes the equation for updating the pheromone values. This is achieved by normalizing the original equation. This creates a pheromone update function that automatically keeps the pheromone values between 0 and 1. This generates a better management of the pheromone when updating its values and facilitating the exploration of solution.

In Max-Min Ant System (*MMAS*) the pheromone values should be set to τ_{min} when they are lower than minimum allowed and τ_{max} when they are larger than maximum allowed. Using this framework on MMAS-HC algorithms, the pheromone values are automatically set between 0 and 1, then according to this τ_{min}=0 and τ_{max}=1. Additionally, there is an explicit relationship between the deposited pheromone and how it is used to build solutions.

4 MMAS-HC for SPSP

The first proposal to solve SPSP with an ACO algorithm was done in [14]. In that paper the author implements an algorithm called Ant Colony System (ACS). In our proposal we use the MMAS algorithm, we additionally incorporate the HC framework to manage the values of pheromones. Our proposal is called Max-Min Ant System with Hyper-cube framework (MMAS-HC). Finally, the heuristics used in our approach are based on the dedication of an employee to a task.

To solve the software project scheduling problem with Max-Min Ant System algorithm and HC framework [10], we have to create a construction graph so that the computational ants can make their routes. We must create the pheromone update rule suitable for HC and establish the heuristic information that will be used to guide the construction of the solutions [22, 23].

The MMAS is one of several ACO algorithms [24, 17]. In MMAS a pheromone range is established, and only one ant is selected by update of the pheromone.

In the overall structure of the proposed algorithm, each task is associated with a set of employees depending on the needs of the tasks, and then it evaluates compliance with the constraints of the proposed solution. If the solution does not violate any constraint then it is a final solution. This can be seen in Fig. 4.

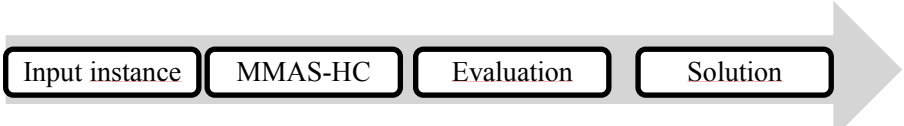

Figure 4 Structure of resolution to MMAS-HC for SPSP

4.1 Graph representation for SPSP

When the computational ants solve a problem, they must make a tour through a construction graph. The tour of an ant from a starting point to an end point determines a solution. The ants choose the paths in the graph according to a probability function. The probabilistic function used by ants is based on the heuristic information from the problem and the memoristic information [19, 20].

Initially, we have to adapt the solution of the problem as a path on a graph so that each task is assigned a group of employees. The degree of dedication of employees to tasks may vary depending on the problem instance solved.

Then the proposed construction graph represents the degree of dedication of the employees to each task in the project. The tasks are selected according to the PG.

The construction graph (CG) has each employee and the possible dedication of a task. The possible dedication is determined by *ded*. This variable is defined in Eq. (13).

$$ded = \frac{1}{(min)} + 1. \tag{13}$$

In Eq. (13) the *min* is the minimum dedication that an employee might have on a task. A representation of the construction graph is presented in Fig. 5, where the *min*=0,25.

For each task we have a CG and it can be represented as a matrix with |E*mp*| number of columns and each column with *ded* rows (nodes per column). A *start* node is created to represent the beginning of the tour (column 0), and then an *end* node is created at the end of the graph to indicate the end of the tour (column |*Emp*| +1).

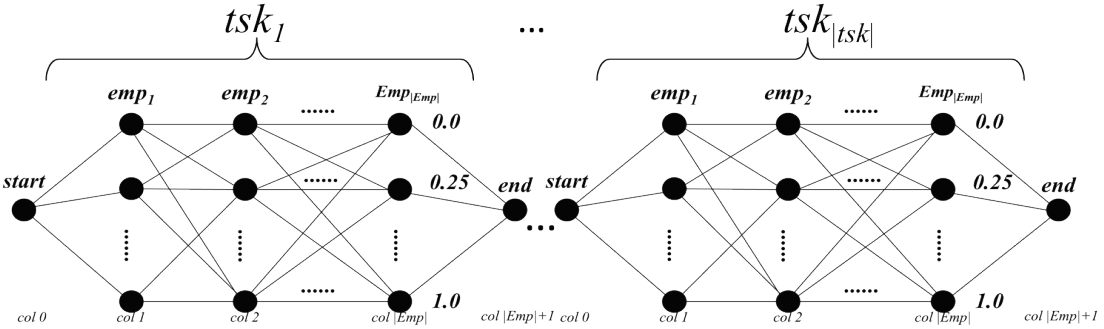

Figure 5 Representation of *CG*= [*ded* × *Emp*] with *min*=0,25 for a set of task

The ants cross the construction graph from a starting point to the ending point, that means by choosing from the *start* node a node from column 1 to column |*emp*+1|. For this the ants use a probability function defined in Eq. (13).

$$p_{ij}^{t} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l=0}^{ded}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, \quad j \in \{1,...,ded\}, \tag{13}$$

where η_{ij} and τ_{ij} are the heuristic information and the pheromone respectively. This probability function determines if the ant travels from node *i* to node *j* in *CG* for task *t*. The influence of the memoristic information is determined by α parameter and the influence of the heuristic information is determined by β parameter. These parameters are determined by bibliographic information or by empirical tests. The heuristic information η_{ij} and the memoristic information τ_{ij} are positive real number, $0<\tau_{ij}<=1$, and $0<\eta_{ij}<1$.

Then the ants follow for the graph choosing a node per column until it reaches the column |*Emp*|. To finish the tour the ants select the *end* node.

To obtain the dedication of each employee, we can take the node *ij* and calculate this as *min***i*, $i \in \{0,.., ded\}$, and $j \in \{1,.., |Emp|\}$.

This cycle must be performed by each ant, for all project tasks. When all ants finish their tour, the assignment of employees to tasks and their dedication to each task will be completed. That means we have a possible solution to the problem, then we need to evaluate its feasibility.

4.2 Pheromone update rules

The Max-Min Ant System is characterized by maintaining the pheromone between a minimum value $\tau_{\min}$ and maximum value $\tau_{\max}$. Moreover the HC framework proposed to maintain the pheromone values between 0 and 1. Thereby by integrating this framework with Max-Min Ant System the values of pheromone will be $\tau_{\min}$=0,0 and $\tau_{\max}$=1,0.

Another important feature in the management of pheromone in Max-Min Ant System algorithms is that only one ant is selected to deposit pheromone for iteration. This ant is the best ant of the iteration. Also we can use the best global ant to deposit pheromone. This ant is the best ant of a generation.

When only one solution is used to reinforce the pheromone trails represented by an ant, it is highly probable that it decreases the exploration of the algorithm, but increases the exploitation of a solution. Thereby when the best ant of the iteration is used, the solutions may vary for iteration. This considerably improves the exploration of the search space. On the other hand, when considering the best global ant the algorithm can converge quickly to a local optimum, increasing the exploitation of a solution, but decreases exploration.

To improve the search for solutions, we need to properly select the best solution of the iteration or the best global solution. This will balance the exploration of new solutions in the search space and exploitation of good

solutions found. Thus, a good alternative is to use a balanced strategy between the best ant of the iteration and the best global ant.

In the management of the pheromone, we must consider the implementation of the artificial evaporation of the pheromone trails on all paths. This is done constantly for iteration or generation according to the evaporation factor.

After the selection of the ant, the ant can deposit pheromone on its path. We must represent both pheromone evaporation and pheromone deposited by the ant using the following formula:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + (1-\rho)\Delta\tau^{upd}, \quad (14)$$

where ρ is a real number and represents the evaporation factor, $0<\rho<1$, when ρ is high, the evaporation of the pheromone is high, on the other hand when ρ is low the evaporation is less. Also we should consider the addition of pheromone dependent $\Delta\tau^{upd}$ and ρ. $\Delta\tau^{upd}$ represents the contribution of pheromone according to the quality of the solution found. If the solution is better, more pheromone is deposited and vice versa [10].To determine the value of $\Delta\tau^{upd}$ we use the fitness function of the problem, this is shown in Eq. 15.

$$\Delta\tau^{upd} = \frac{((wp^{\cos} pjt^{\cos} + wp^{len} pjt^{len})^{upd})^{-1}}{\sum_{h=1}^{m}((wp^{\cos} pjt^{\cos} + wp^{len} pjt^{len})^{h})^{-1}}, \quad (15)$$

where $(wp^{cos}pjt^{cos} + wp^{len}pjt^{len})^{upd}$ is the fitness function of the best ant used to update the pheromone, where wp^{cos} and wp^{len} represent the importance of pjt^{cos} and pjt^{len} respectively. $(wp^{cos}pjt^{cos} + wp^{len}pjt^{len})^{h}$ this represents the quality of the solutions of the other ants not used to update the pheromone.

4.3 Selection of a heuristic

To build good solutions, the ACO algorithms use a type of information called "heuristic information" directly related to the problem, and more specifically to the fulfilment of the goal of the problem. The heuristic information can guide the behaviour of ants to achieve better solutions. In fact, without heuristic information, the ants only are guided by the pheromone as a greedy algorithm.

We propose to use as heuristic information the degree of occupation that the employee has. Thereby if an employee was assigned to other tasks previously, the employee has a lower probability of being assigned to the current task. We use an array $he[i]$ to represent the heuristic information of the employee to select the node i, $he[1...ded]$, The heuristic information can be obtained using the Eq. (16).

$$he[i] = \begin{cases} \frac{t[ded-i-1]}{s}, & if\ occup[k] > 0{,}5 \\ \frac{t[i]}{s}, & else \end{cases}, \quad (16)$$

where s is the sum of array t defined in Eq. (18), and $t[1\ldots ded]$ is a temporary array defined as follow:

$$t[i] = \begin{cases} d[i] + occup[k] - 0.5, & if\ occup[k] > 0{,}5 \\ d[i] + occup[k], & else \end{cases}. \quad (17)$$

The array $d[1..ded]$ is dedication for possible nodes to select. The array $occup[1\ldots|Emp|]$ is the degree of occupation that has the employee.

$$sum = \sum_{i=0}^{ded} tmp[i]. \quad (18)$$

We can also consider a static heuristics, based on employee salaries. In this case we would select employees with lower salaries to reduce the final costs of the project.

4.4 Algorithm

We represent the algorithm using a flow chart to better understand their behaviour. This algorithm is a modification of that proposed in [24]. We propose a MMAS and the implementation of HC framework. This modification allows us to use only one solution for update of the pheromone. Furthermore the pheromone values are between 0 and 1. The algorithm can be described in the following flow diagram:

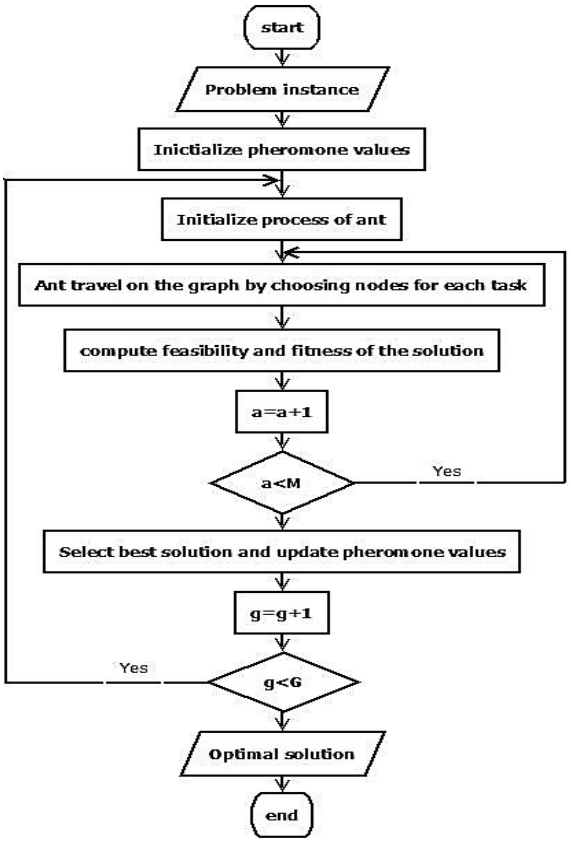

Figure 6 A flowchart describing the MMAS-HC algorithm

First, the problem instance is loaded. The information of the precedence of tasks, employees, and skills is used to populate the corresponding data structures. Then the algorithm is ready to begin its iterative activities. The algorithm may iterate until a certain number of iterations or time is met. The steps of the algorithm are as follows:

- After loading the instance of the problem and the data is stored in data structures, the algorithm sets the values of pheromone to a minimum value τ_{min}.
- Then the ant process is initialized and the colony size is established.
- Every ant in the colony performs the following activities: The ant travels through a construction graph for each task. When an ant completes a tour a possible solution is obtained. Then the feasibility of the generated solution is computed. In addition the fitness of the solution is obtained.
- When all ants in the colony complete their tour, the best solution is selected. Then the pheromone is deposited on the path to the best solution found.
- Finally all generations end and an optimal solution is obtained.

5 Results

In order to evaluate empirically the performance of our approach, we have done the comparison of a preliminary version of our algorithm with the results presented in [14]. The algorithms used are compared with the instances created by the generator of instances proposed in [16].The results are presented in Tab. 2.

Table 2 Fitness comparison of MMAS-HC and ACS for SPSP

Instance with	**MMAS-HC**	**ACS**
|*skl*|= 2-3	*fitness*	*fitness*
5 |*emp*|, 10 |*tsk*|	**3,311**	3,558
10 |*emp*|, 10 |*tsk*|	**2,617**	2,638
15 |*emp*|, 10 |*tsk*|	**1,996**	2,083
20 |*emp*|, 10 |*tsk*|	1,892	**1,858**
10 |*emp*|, 20 |*tsk*|	**6,211**	6,369

The fitness corresponds to a minimization problem. From this point a lower fitness represents a better solution. All instances used have |skill|=2-3 for the tasks and the employees. The number of employees is 5, 10, 15, and 20. And the number of tasks is 10, 20.

Tab. 2 present the fitness obtained with our proposal MMAS-HC and ACS algorithm. The results presented show that our proposal obtains better results in most cases. When instances have a number of employee |*emp*| = 20, the best result is obtained with ACS. These are only preliminary results, since still the proposals can be improved and this can be tested with more extensive number of instances.

6 Conclusion

We have presented a model to the resolution of the SPSP using an MMAS algorithm and HC framework, which provides a well-defined structure with *n*-dimensional space for the memoristic information determined by the pheromone. The SPSP is an important problem for the software project management and its resolution is very complex. We define the structures, graph construction, and appropriate heuristic information to solve the SPSP by MMAS-HC algorithm. In addition we define appropriate fitness function to evaluate quality of solutions. We believe that ACO is promising attacking this problem. We demonstrated that SPSP can be represented as a graph based search problem, and ACO can handle this kind of problem successfully.

In the future, we hope to test our proposal with other more complex benchmarks and we hope to improve the components of the framework.

Acknowledgements

Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1140897. Ricardo Soto is supported by Grant CONICYT/FONDECYT/ INICIACION/11130459. Fernando Paredes is supported by Grant CONICYT/FONDECYT/REGULAR/1130455. Franklin Johnson is supported by Postgraduate Grant PUCV 2015.

7 References

[1] Nan, N.; Harter, D.E. Impact of Budget and Schedule Pressure on Software Development Cycle Time and Effort. // IEEE Transaction on Software Engineering. 35, 5(2009), pp. 624-637. DOI: 10.1109/TSE.2009.18

[2] Brucker, P; Drexl, A.; Mohring, R.; Neumann, K.; Pesch, E. Resource-Constrained Project Scheduling: Notation, Classification, Models and Methods. // European Journal of Operational Research. 112, (1999), pp. 3-41. DOI: 10.1016/S0377-2217(98)00204-5

[3] Kolisch, R. Efficient priority rules for the resource-constrained project scheduling problem. // Journal of Operations Management. 14, 3(1996), pp. 179-192. DOI: 10.1016/0272-6963(95)00032-1

[4] Gomes, H.; De Assis das Neves, F.; Souza M. Multi-objetive metaheuristic algorithms for the resource-constrained project scheduling problem with precedence relations. // Computer & Operation Research. 44, (2014), pp. 92-104. DOI: 10.1016/j.cor.2013.11.002

[5] Liao, T. W.; Egbelu, P.; Sarker, B.; Leu, S. Metaheuristics for project and construction management a state-of-the-art review. // Automation in Construction. 20, 5(2011), pp. 491-505. DOI: 10.1016/j.autcon.2010.12.006

[6] Chang, C. K.; Jiang, H.; Di, Y.; Zhu, D.; Ge, Y. Time-Line Based Model for Software Project Scheduling with Genetic Algorithms. // Information and Software Technology. 50, (2008), pp. 1142-1154. DOI: 10.1016/j.infsof.2008.03.002

[7] Chang, C. K.; Christensen, M. J; Zhang, T. Genetic Algorithms for Project Management. // Annals of Software Engineering. 11, (2001), pp. 107-139. DOI: 10.1023/A:1012543203763

[8] Cyril, C.; Hephzibah, D. Adaptive Task Scheduling Based on Multi Criterion Ant Colony Optimization in Computational Grids. // ICRTIT, (2012), pp. 99-102.

[9] Blum, C.; Dorigo, M. HC-ACO: the hyper-cube framework for ant colony optimization. // Proceedings MIC2001 Metaheuristics Int. Conf., (2001), pp. 399-403.

[10] Johnson, F.; Crawford, B.; Palma, W. Hypercube framework for ACO applied to timetabling. // Artificial Intelligence in Theory and Practice, IFIP 19WCC, 217, (2006), pp. 237-246.

[11] Ozdamar, L.; Ulusoy, G. A survey on the resource-constrained project scheduling problem. // IIE Transactions, 27, 5(1995), pp. 574-586. DOI: 10.1080/07408179508936773

[12] Gutjahr, W.; Rauner, S. An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria // Original Research Article Computers & Operations Research. 34, 3(2007), pp. 642-666. DOI: 10.1016/j.cor.2005.03.018

[13] Zhang, L.; Wu, Z.; Wang, K.; Wang, W. Application of Improved Ant Colony Algorithm to JSP. // College of Mechanical & Material Engineering, China Three Gorges

Univ. Natural Sciences, Yichang 443002, China, 4, (2009), pp. 75-78.
[14] Xiao, J.; Ao, X. T.; Tang, Y. Solving software project scheduling problems with ant colony optimization. // Computers and Operations Research. 40, 1(2013), pp. 33-46. DOI: 10.1016/j.cor.2012.05.007
[15] Barreto, A.; Barros, M. de O.; Werner, C. M. L. Staffing a software project: A constraint satisfaction and optimization-based approach. // Computers and Operations Research. 35, 10(2008), pp. 3073-3089. DOI: 10.1016/j.cor.2007.01.010
[16] Alba, E.; Chicano, F. Software project management with Gas. // Information Science, 177, 11(2007), pp. 2380-2401.
[17] Rubio, J. M.; Crawford, B.; Johnson, F. Solving the university course timetabling problem by hypercube framework for ACO. // J. Cordeiro and J. Filipe, editors, ICEIS (2), (2008), pp. 531-534.
[18] Dorigo, M.; Stutzle, T. Ant Colony Optimization. MIT Press, USA, 2004. DOI: 10.1007/b99492
[19] Berrichi, A.; Yalaoui, F.; Amodeo, L.; Mezghiche, M. Bi-objective ant colony optimization approach to optimize production and maintenance scheduling. // Computers and Operations Research. 37(9), (2010), pp. 1584-1596. DOI: 10.1016/j.cor.2009.11.017
[20] Dorigo, M.; Maniezzo, V.; Colorni, A. The ant System: Optimization by a colony of cooperating agents. // IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics. 26, 1(1996), pp. 29-41. DOI: 10.1109/3477.484436
[21] Dorigo, M.; Gambardella, L. M. Ant colony system: A cooperative learning approach to the traveling salesman problem. // IEEE Transactions on Evolutionary Computation. 1, 1(1997), pp. 53-66. DOI: 10.1109/4235.585892
[22] Chen, W.; Zhang, J. Ant colony optimization for software project scheduling and staffing with an event-based scheduler. // Software Engineering, IEEE Transactions on. 39, 1(2013), pp. 1-17. DOI: 10.1109/TSE.2012.17
[23] Abdallah, H.; Emara, M.; Dorrah, H. T.; Bahgat, A. Using ant colony optimization algorithm for solving project management problems. // Expert Systems with Applications. 36, 6(2009), pp. 10004-10015. DOI: 10.1016/j.eswa.2008.12.064
[24] Stützle, T.; Hoos, H. H. MAX-MIN Ant System. // Future Generation Computer Systems. 16, 8(2000), pp. 889-914. DOI: 10.1016/S0167-739X(00)00043-1
[25] Dorigo, M.; Di Caro, G. Ant colony optimization: a new meta-heuristic. // Evolutionary Computation, CEC 99. Proceedings of the 1999 Congress, Vol. 2, (1999), pp. 1453-1477.

Authors' addresses

Broderick Crawford, Ph.D., Professor
Pontificia Universidad Católica de Valparaíso
Avenida Brasil 2950, Valparaíso, Chile
and
Universidad San Sebastián
Bellavista 7, Recoleta, Chile
and
Universidad Central de Chile
Calle Toesca 1783, Santiago, Chile
E-mail: broderick.crawford@ucv.cl

Ricardo Soto, Ph.D., Professor
Pontificia Universidad Católica de Valparaíso
Avenida Brasil 2950, Valparaíso, Chile
and
Universidad Autónoma de Chile
Pedro de Valdivia 641, Santiago, Chile
E-mail: ricardo.soto@ucv.cl
and
Universidad Cientifica del Sur
General Suarez 287, Lima, Perú
E-mail: ricardo.soto@ucv.cl

Franklin Johnson, Professor
Pontificia Universidad Católica de Valparaíso
Avenida Brasil 2950, Valparaíso, Chile
and
Universidad de Playa Ancha
Av. Leopoldo Carvallo 270, Valparaíso, Chile
E-mail: franklin.johnson@upla.cl

Sanjay Misra, Ph.D., Professor
Department of Computer and Information Science, Covenant University, Nigeria
and
Atilim University, 06836 - Incek, Ankara Turkey
E-mail: smisra@atilim.edu.ng

Fernando Paredes,Ph.D., Professor
Escuela de Ingeniería Industrial, Universidad Diego Portales,
Manuel Rodríguez Sur 415, Santiago, Chile
E-mail: fernando.paredes@udp.cl

Eduardo Olguín, Professor
Universidad San Sebastián
Bellavista 7, Recoleta, Santiago, Chile
E-mail: eduardo.olguin@uss.cl