TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 6, December 2021, pp. 1820~1829
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i6. 16758

Solving software project scheduling problem using grey wolf optimization

Marrwa Abd-AlKareem Alabajee, Dena Rafaa Ahmed, Taghreed Riyadh Alreffaee
Department of Software, College of Computer Science & Mathematics, University of Mosul, Mosul, Iraq

Article Info

Article history:

Received May 20, 2020
Revised Oct 2, 2021
Accepted Oct 14, 2021

Keywords:

Grey wolf optimization
Resource-constrained project scheduling
Software project management
Software project scheduling problem

ABSTRACT

In this paper, we will explore the application of grey wolf optimization (GWO) methodology in order to solve the software project scheduling problem (SPSP) to seek an optimum solution via applying different instances from two datasets. We will focus on the effects of the quantity of employees as well as the number of tasks which will be accomplished. We concluded that increasing employee number will decrease the project's duration, but we could not find any explanation for the cost values for all instances that studied. Also, we concluded that, when increasing the number of the tasks, both the cost and duration will be increased. The results will compare with a max-min ant system hyper cube framework (MMAS-HC), intelligent water drops algorithm (IWD), firefly algorithm (FA), ant colony optimization (ACO), intelligent water drop algorithm standard version (IWDSTD), and intelligent water drop autonomous search (IWDAS). According to these study and comparisons, we would like to say that GWO algorithm is a better optimizing tool for all instances, except one instance that FA is outperform the GWO.

Corresponding Author:

Marrwa Abd-AlKareem Alabajee
Department of Software
Mosul University
Al Majmoaa Street, Mosul, Iraq
Email: marrwa_zedan@uomosul.edu.iq

1. INTRODUCTION

All of us know that software engineering is a discipline of engineering which is interested in all aspects of software production. It is also the systematic approach which is utilized in the software engineering. Sometimes, the applications of business are not necessary in this manner. There is a new software of business which is often developed via expanding as well as changing current systems or via integrating and configuring off-the-shelf software of the system [1].

There are many different processes of the software, but each one must involve four activities, for this reason they represent the fundamental of the software engineering which contains: software specification, Software design and implementation, software validation, and software evolution. A process model of the software is a simple explaination of software process. Every one of them was showing a process from a specific perspective. The process model of the software involves many models such as:

- The waterfall model: This model takes the essential process of development, specification, evolution and validation. It also represents activites as separate stages of the process.
- Incremental development: The system of this model will be developed as a series of increments (versions), the increamental model interleaves the activities of specification, development, and validation.

- Reuse-oriented software engineering: The development process of the system is focusing on merging reusable components into a system instead of evolving them from scratch because it based on the presence of a significant number of these reuseble components.
- Agile software approaches which incorporate the activities such as: testing and the elicitation of the requirements into the implementation and the design. These approaches take the design and implemation into consideration with main activities in the pocess of the software.

The development of the software is a very complex domain which faces many problems through the development process [2]. The late delivery of the software projects is one of these problems. However, several reasons are lead to late the delivery of the software projects. Such as: modifying the requirement of the customer, issues of technical, miscommunication between the members of the team, inappropriate schedules of the time, failure to analyze the risks involved, and resources underestimation such as man-power, effort, and cost. Since with modifying times the complexity, software size have also modified. Analyze the requirement of the software and review for its feasibility is very essential issue [3].

Whatever the process models of the software are utilized, project management of the software is an essential portion of the software engineering. The projects of the software require to be managed because the professional software engineering is always follows the schedule constraints and budget of the organization. The job of the project manager is to guarantee that the project of the software meets and beats these constraints as well as delivering software with high-quality [1].

It is necessary for the managers of the projects to planning to: nail the project development deadlines, scheduling and estimating the development of the project, and specifying people to tasks. The managers monitor the work to guarantee that it is accomplished to the standards which required, and supervise the progress to examine that the development of the project is within budget and time [1], [4]. In general, the management of the project composes of five phases: initiating, planning, executing, controlling, and closing. When it comes to the development of the software project, scheduling of the project, planning of the project, monitoring, risk management and controlling tasks will be taken into account [5].

The scheduling of the software project starts with analyzing the requirement of the software and on the basis of this software estimation, size is made. The software project is then split into smaller parts which are known as tasks. The network of the task assigns the various tasks, their expected period of completion and different tasks dependency on each other. Next stage consists of resources estimating which needed for tasks completion. The team of the project is decided for each task. A timeline for each task is assigned and it is essential to keep a check on whether the development of the software is going along the assigned limit of the time thereby observating and controlling the development process of the software [3]. The software project scheduling problem (SPSP) is a spesific situation of the project scheduling problems (PSP). The problem of SPSP is employees are assigned to tasks with the intent of reducing the period as well as the cost of a project with the consideration of the resource constraints and the tasks precedence [6].

The SPSP is associated with the resource constrained project scheduling (RCPS), some variations between SPSP and RCPS are: first, in SPSP there is a cost related to the employees and a cost related to a project that must be reduced (in addition to the project period). Added to these in RCPS there are many types of resources which have been evaluated, among renewable, nonrenewable and doubly constrained resources. While the employee with several possible skills is the only kind in the SPSP. We must observe that the skills of SPSP are variant from resource types of RCPS. In addition, every action in the RCPS needs different quantities of every resource while skills of SPSP are not quantifiable entities [7], [8]. SPSP is considered as a combinatorial optimization problem (COP) which has several different possible allocations between tasks and employees and every allocation has different time as well as cost. The SPSP has been resolved using incomplete techniques (metaheuristics and heuristics appraoches) which check only possible allocations parts in order to find near optimal solutions in an acceptable time and effort [9].

In this topic, i would like to say that in the last years, a large amount of research has been performed. The studies that have been conducted have focused on different sides [10]:

- Suggesting models in order to resolve the problem of assigning employees to tasks so that the price and the period of the project are reduced.
- Applying or inserting more efficient algorithms of optimization.
- Studying the various metaheuristics performance to see which one is more acceptable to resolve the problem.

The grey wolf optimization (GWO) is an algorithm of meta-heuristic which inspired by grey wolves. The GWO algorithm mimics the leadership hierarchy and hunting mechanism of grey wolves in nature [11]. In this paper, the grey wolf optimization (GWO) algorithm has been used to solve the problem of software project scheduling (SPSP) and find effective assigning employees to tasks so that the cost and period of the project are reduced. Added to these the results of GWO algorithm have been compared with other techniques.

Many research articles have been presented in the software project scheduling problem field. The variant in the gained outcomes is the result of utilizing a various methodology in each of these articles to cover most of the previous work presented in this field. Alreffaee and Alabajee [12] used Whale optimization algorithm (WOA) for solving the problem of SPS and utilized it on various instances from three datasets. When the datasets have a few tasks the WOA gave good results but it failed to find feasible solutions when increasing task's number. Crawford *et al.* [13] presented the self-configuring of the velocity parameter of IWD metaheuristic that effects the algorithm behaviour by having a direct relationship between removed soil and velocity. The proposed methodology was used in solving the SPSP and the outcomes of standard version (IWDSTD) and configuring version (IWDAS) were very similar.

Almshhadany and Ibrahim [9] introduced a new algorithm of swarm intelligence (multi-objective artificial fish) based on the Pareto archived evolution strategy (PAES) algorithm principles and it was used to solve the problem of SPSP. At the same years, Crawford *et al.* [6] proposed metaheuristic intelligent water drops for solving the SPSP optimization problems, it obtained a good result. The outcoms were compared with an ant colony optimization (ACO) approach which is known as ACS-SPSP algorithm and with SPSP utilizing a max-min ant system with a hyper-cube framework (MMAS-HC). Velázquez's *et al*. [10] presented a survey on the existing articles that focused on resolving the SPSP problem. This paper analyzed and categorized a number of research studies which take into account a group of criteria namely, the problem model, the goals, and the techniques of the optimization which utilized to resolve the problem, the methodology was utilized to evaluate the main outcomes and the different methods.

Rachman and Ma'sum [14] compared the max-min ant colony system performance with ant colony extended algorithm to solving the problem of SPSP with the assistance of fitness value, and ant colony extended algorithm was show a better performance. Also in the same year, Nigar [15] gave a new formula to SPSP as an optimization problem under uncertainties and dynamics for hybrid scRUmP software model. The mathematical model has four different objectives: project duration, cost, robustness, task fragmentation and stability. Crawford *et al.* [16] explored the firefly algorithm as the first use to solving SPSP problem, to prove the firefly algorithm soundness and viability, the outcomes were compared with other techniques such as (genetic algorithm (GA) and ant colony algorithm (ACO)).

Biju *et al*. [5] presented a differential evolution (DE) method to solve the SPSP problem. The proposed method superiority was demonstrated and experimented by solving the SPSP on 50 random instances and the outcomes were compared with some previous techniques. Also, Crawford *et al.* [17] to solve the problem of SPSP they design a model using ACO algorithm with the employ of Hyper-Cube framework. This allows to autonomously handling the exploration of search area to attaining hopeful solutions. Lately, Luna *et al.* [18] analyzed the scalability of eight multiobjective algorithms when they were applied to the SPSP problem, they have performed an experimental evaluation utilizing a benchmark of 36 automatically generated instances with increasing size. The PAES algorithm proved that it is not only the algorithm which scales the best but also shown the best solution quality (in terms of HV). Also in the same year, Hanchate and Bichkarto [19] solved the SPSP problem by presented a particle swarm optimization algorithm (PSO). The effect of PSO parameter on the cost and time of the project was studied and some better outcomes were obtained in terms of minimum cost estimation and time of the software as compared to some techniques (GA and ACO).

In the same year, Crawford *et al.* [20] proposed a max-min ant system algorithm as a new approach to solve the SPSP problem. The outcomes were compared with ACO System and GA. The proposed method for smaller instances gave the best results. Last but not least Xiao *et al.* [21] employed ACO approach in order to solve the problem of SPSP. To consider the task efforts factors, task importance, and allocated dedications of employees, six domain-based heuristics were designed. Experimental outcomes of ACS-SPSP on 30 random instances obtained higher hit rates with more accuracy when compared to (GA) solution.

2. RESEARCH METHOD

The SPSP finds the table of employee-task, with the considration of task precedence and resource constraint. As well as the project's duration and cost should be minimized. This section includes the description of software project scheduling problem (SPSP). In addition, it includes an explanation of all steps of GWO algorithm.

2.1. The formulation's problem of the SPSP

SPSP aim is to get the perfect and correct assigning of the employee to tasks of the project. SPSP should take into account the employees' rewards and skills in determining the needs of each task. In order to schedule any project of the software, we need information about: the project tasks, the employees who must

work on various tasks and the skills which needed to complete the different tasks of software project. We described this information in detailed and then explained the mathematical models [6], [9]:

– Tasks

All the major activities that must do to plenary a project are project tasks. These activities may contain the analysis, design of different components, programming, documentation and the entire testing process. Each project has tasks (Ta), every task (ta_i) has a set of skills $(ta_i^{skills})$and an estimated effort (ta_i^{effort}). TPG is the Precedence Graph of the task, it used to observe task precedence. This graph is non occuring in cycle, knowed as G (E,V). The set of tasks represented by $V = \{ta_i, \dots, ta_{|T|}\}$. Set of edges is the linkage between the tasks, the edge $(ta_i, ta_j) \in E$ indicates that the task ta_i is come before the task ta_j. So, project tasks is knowed as: $ta = \{ta_1, \dots, ta_{|T|}\}$, |T| refers to the tasks number [6].

– Skills

Various skills need to accomplished the different tasks that staffs must have to be allocated to those tasks, e.g: leadership, expert analyst, expert in some programming language, GUI designer, database expert, interfaces expert, leader and others. To complete the tasks, it should has a group of skills. The set of all skills should be in the project is konwn $asski = \{ski_1, \dots, ski_{|S|}\}$, where |S| is the maximum of skills [6], [17].

– Employees

Employees are the only source of SPSP. They have the skills of software engineering. The set of employees are allocated to tasks which defined as $empl = \{empl_1, \dots, empl_{|EM|}\}$, whereas |EM| is the employees' maximum number. The emloyees must comply with the task's skills, every employee ($empl_i$) has a group of skills ($empl_i^{skills}$) which is a subset of all various skills available in the staff members (S) of the project, each employee also has a salary($empl_i^{salary}$), and a maximum dedication($empl_i^{max}$). It is a number of hours that the employee work on the project ($empl_i^{max} \in [0,1]$). The SPSP solution is illustrated as a matrix D of size ($EM * T$) where every D_{ij} is the amount of employee's dedication ($empl_i$) to task (ta_j), Table 1 explains the solution for matrix D.

To avoid overwork, the dedication's summation for each employee must not exceed the maximum dedication of employee. The amount of maximum dedication for all employees is always assumed to be 1. The aim of SPSP is to reduce both cost and completion time of the project [6], [9], [17].

Table 1. Example of solution for matrix D

D_{ij}	ta_1	ta_2	ta_3	ta_4	ta_5	ta_6
*empl*1	0.00	0.50	0.00	1.00	0.25	0.00
*empl*2	0.25	0.00	0.75	0.00	0.50	1.00
*empl*3	1.00	0.00	0.25	0.25	0.00	0.00
*empl*4	0.00	0.75	0.00	0.25	0.00	0.50

To calculate the duration of project, we have to get each tasks duration ta_j^{dur}. For this reason, we utilize the effort in month ta_j^{effort} and the totality of the dedication of every employee allocate to the task j. As the following [6]:

$$ta_j^{dur} = \frac{ta_j^{effort}}{\sum_{i=1}^{EM} D_{ij}} \quad (1)$$

with the task's duration and the information of the precedence in TPG, we can get the starting time ta_j^{start} and the ending time ta_j^{end} for task j.

We must look at two cases:

– First: The staring time is ($ta_j^{start} = 0.0$).

– Second: The starting time is equal to the ending time ta_c^{end} when the task has precedence, ta_c^{end} is the ending of the time of the maximum overdue previous task. ta_j^{start} is known in (2) [6]:

$$ta_j^{start} = \begin{cases} 0 \; if \; \forall \, c \neq j, (ta_c, ta_j) \notin E \\ \max\{ta_c^{end} | (ta_c, ta_j) \in E \end{cases} \quad (2)$$

The finishing time for a task j (ta_j^{end}) is calculated from (3) [9]:

$$ta_j^{end} = ta_j^{start} + ta_j^{dur} \quad (3)$$

To compute the total project duration, we utilize the information of the Gantt chart and the duration of every task in the project. For this, we just need the ending time of the last task, the whole duration is calculated from (4) [6]:

$$pro_{dur} = max\{ta_l^{end} | \forall\ l\ \neq j, (ta_j, ta_l) \notin E\} \quad (4)$$

In the project, each task's cost must be previously obtained, for calculating later the whole cost of the project($\mathrm{pro_{cost}}$), for the task j, the cost ($\mathrm{ta_j^{cost}}$) can be computed utilizing the (5), Then The project's cost is the cost summation of all tasks in the project as illustrated in (6) [6], [9]:

$$ta_j^{cost} = \sum_{i=1}^{EM} empl_i^{salary} * D_{ij} * ta_j^{dur} \quad (5)$$

$$pro_{cost} = \sum_{j=1}^{T} ta_j^{cost} \quad (6)$$

Usually, we must define a fitness function to meet the aim of the problem, The SPSP fitness is related to the whole duration $\mathrm{pro_{dur}}$ and whole cost $\mathrm{pro_{cost}}$ of the project. The fitness function is given in (7) [17].

$$f(X) = pro_{cost} * W_{cost} + pro_{dur} * W_{dur} \quad (7)$$

Where the ($\mathrm{W_{cost}}$) and ($\mathrm{W_{dur}}$) are real values. It is essential to introduce a few constraints and limitation into the project in order to ensure that a feasible and realistic SPSP optimization schedule is achieved, for more see [6], [7], [9].

2.2. Summarized description of GWO algorithm

There are several forms of intelligence which emerge from the insects' community, fish, birds, or mammals [22], one of the recently suggested swarm intelligence-based algorithms is the grey wolf optimization. This algorithm was developed in Mirjalili *et al.* [11]. It is inspired by grey wolves in nature that searching for the optimal way for hunting preys [23].

2.2.1. Basics of GWO

After studying the problem of scheduling in details. GWO algorithm will be employed to solve this problem. This subsection presents in briefly the mathematical model of GWO algorithm, including social hierarchy, tracking, encircling and attacking prey of wolves. The GWO algorithm for SPSP is outlined.

– Social hierarchy

Alpha, beta and delta are considered as the fittest solution, second and third best solutions as respectively, omega are the rest of the candidate solutions. The optimization process in the GWO algorithm is guided by α, β, and δ. The ω wolves follow these three wolves [24].

– Encircle prey

When the wolves do the hunting they will encircle the prey. This is represented in (8) and (9) respectively [25]:

$$\vec{D} = |\vec{C} . \vec{X}_p\ (t) - \vec{X}\ (t)| \quad (8)$$

$$\vec{X}\ (t+1) = \vec{X}_p\ (t) - \vec{A} . \vec{D} \quad (9)$$

where:

t : The current iteration.

$\vec{\mathrm{A}}$, $\vec{\mathrm{C}}$: The coefficient vectors of the prey.

$\vec{\mathrm{X}}_\mathrm{p}(\mathrm{t})$: Prey position.

$\vec{\mathrm{X}}(\mathrm{t})$: Grey wolves position.

The vectors $\vec{\mathrm{A}}$ and $\vec{\mathrm{C}}$ are calculated as follows in (10) and (11) [26].

$$\vec{A} = 2\vec{a} . \vec{r_1} - \vec{a} \quad (10)$$

$$\vec{C} = 2 . \overrightarrow{r_2} \quad (11)$$

where:
$\vec{a}$ elements will linearly decreased from 2 to 0 over the course of iteration.
$\overrightarrow{r_1}$, $\overrightarrow{r_2}$: are random vectors in between [0, 1].

– Hunting

The mechanism of the hunt depends on the position of the superior search agents so it's guided by alpha, Beta and delta [25]. The equations sets in from 12 to 18 signify the updates of the position [27]:

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1} . \overrightarrow{X_\alpha} - \vec{X}\right| \quad (12)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} . \left(\overrightarrow{D_\alpha}\right) \quad (13)$$

$$\overrightarrow{D_\beta} = \left|\overrightarrow{C_2} . \overrightarrow{X_\beta} - \vec{X}\right| \quad (14)$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} . \left(\overrightarrow{D_\beta}\right) \quad (15)$$

$$\overrightarrow{D_\delta} = \left|\overrightarrow{C_3} . \overrightarrow{X_\delta} - \vec{X}\right| \quad (16)$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} . \left(\overrightarrow{D_\delta}\right) \quad (17)$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \quad (18)$$

– Attacking prey (exploitation)

The gray wolves finish the hunt by attacking the hunt as soon as it stops. So, the value of $\vec{a}$ is decreased. The fluctuation range of $\vec{A}$ also decreases by $\vec{a}$. $\vec{A}$ is a random value in [-a,a] where a is reduced from 2 to 0 through iterations, the following place of a search agent can be in any place between its existing place and the place of the victim when random values of $\vec{A}$ are in [-1,1] [24], [28].

– Search for prey (exploration)

Alpha, beta and delta, they splay from each other to seek and huddle to offense the victim. Here $\vec{A}$ is exercised as maximal than 1 or minimal than -1 to point the search agent to space and meet the victim. This allow GWO search globally and asserts exploration [25].

2.2.2. The GWO algorithm for SPSP

This section displays the steps of GWO-SPSP, which employed the GWO algorithm in solving SPSP:

– Reading the instances file which was created through the generator of the SPSP instance. All data's types of SPSP are modelling which saved in this file, including number of task, number of employee, TPG of the project, group of needed skills of a task, the dedication required for every task, skills of each employee, employees' salaries and so on, and then the SPSP model was structured according to pervious data.
– Initializing the parameters of the GWO algorithm such as A, C, a, maximum number of cycles and agents number.
– Generating the initial random population as schedules, each schedule has two dimensions (employee and task).
– Calculating the duration and cost of every search agent, then calculate the fitness function and determine the values of X_α, X_β , X_δ : namely: the best, the second and the third best search agent respectively.
– Repeating the steps (6-9) when the stopping condition is not satisfied.
– For every search agent, update the position of current search agent.
– Update the GWO algorithm parameters: a, A, and C.
– Calculating the fitness of every search agent after updating the positions of grey wolf population,
– UpdateX_α, X_β andX_δ.
– Return X_α which contains best schedule of the total project.

3. RESULTS AND ANALYSIS

The experiments were carried out on a computer with Intel Core i5 and Windows7 Ultimate was built using matlab 2017. GWO-SPSP runs every experiment for 30 trials using the parameters' values which

presented in Table 2. Table 2 shows the values of parameters which selected based on experience. The average values were used to evaluate the quality of solution.

Table 2. Values of parameters

Name	Value
Number of cycle	1000
Number of agent	40
W_{cost}	10^{-6}
W_{dur}	10^{-1}

3.1. Description of datasets

To implement a significative study, we used a different instance from two dataset that are generated by the instance generator, an instance generator is an easily parameterizable program which derive automatically instances for a problem, a deeper description of the generator can be searched at URL http://tracer.lcc.uma.es/problems/psp [7]. DS1 (Dataset1): The skills' amount is 10; the number of the skills of an employee is 2–3; the number of required skills in the task is 2–3, DS1 is consists of nine instances Its acombinations of employee (5,10,15), Task (10,20,30) and skill (10), the first instance is (5 employee, 10 task, 10 skill), second instance (5 employee, 20 task, 10 skill), third instance (5 employee, 30 task, 10 skill) and so on.

DS2 (Dataset2): The skills' amount is 5; the number of skills of an employee is 2–3; the number of required skills in the task is 2–3. DS2 is also consists of nine instances Its acombinations of employee (5,10,15), Task (10,20,30) and skill (5). the first instance is (5 employee, 10 task, 5 skill), second instance (5 employee, 20 task, 5 skill), third instance (5 employee, 30 task, 5 skill) and so on.

The instances in DS1and DS2 have the same TPG; the tasks have the same effort and each employee has the same values of maximum dedication and salary. We have written a program that analyzes and interprets data for each set of data sets that taken from the instance generator and storing the outputs in another file that we used later with scheduling issue files and gray wolf algorithm files to solve the optimization issue, to find the best solution for them in the least time and cost.

3.2. The effect of employees' number

If we want to study the effect of employees' number and tasks' number on the outcomes, we should take a various instance from DS1, then should note that human resources (employees) are the only resources for this problem and the tasks duration is directly associated with the utilize of this resource. The task's duration differ according to employees allocated to that task, it is essential to show that this interaction can vary the duration of the project. As shown in Table 3, we used three different numbers of employees: 5, 10 and 15. The results show that when the number of employees is increased, the duration of the project decreases. But with the cost of the project, we cannot discover a direct relationship with it.

The convergence curve to better solution for three instances that are taken from DS1 is shown in Figure 1, the numbers of tasks are fixed only the number of employees is variable with five skills. For the three instances, we can obtain better solutions from iteration (100) and converges slowly to the best solution.

On the other hand, the influence of the different number of employees on the three instances from DS2 was also studied. From Table 4 we concluded that when the number of employees increasing, this will decrease the duration and cost of the total project. The convergence curve of DS2 is shown in Figure 2, the algorithm gives better solutions for instance (5 employees) and instance (15 employees) from iteration 200, and from iteration 300 for instance (10 employees), the curve converges slowly to the best solution for the three instances that studies from DS2 where the numbers of tasks are constant only the number of employees are variable with ten skills.

Also, it was observed that increasing the level of skill set also increases the search space so when comparing the outcomes of Table 3 and Table 4 we concluded that increasing the number of skills will affect on the search space size, it becomes bigger and this does not assistance the search operation, so the fitness function of DS2 instances is higher than the fitness function of DS1 instances.

Table 3. Results of DS1 with (5, 10, 15) employees' number

Instance	Fitness	Cost	Duration
5_10_5 (employee-task-skill)	2.1191	791280	13.2783
10_10_5 (employee-task-skill)	1.4086	799900	6.0875
15_10_5 (employee-task-skill)	1.2911	776020	5.1508

Table 4. Results of DS2 with (5, 10, 15) employees' number

Instance	Fitness	Cost	Duration
5_10_10 (employee-task-skill)	3.0275	940600	20.8694
10_10_10 (employee-task-skill)	2.2432	928110	13.1507
15_10_10 (employee-task-skill)	1.8646	717080	11.4757

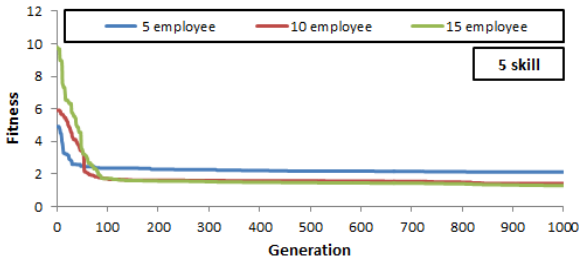

Figure 1. Convergence curve of DS1

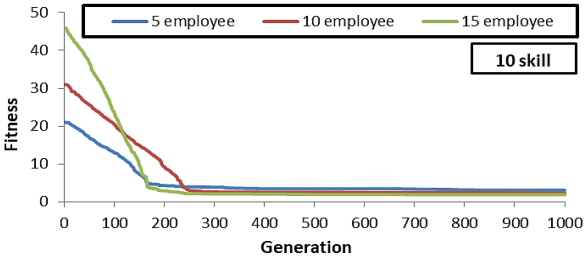

Figure 2. Convergence curve of DS2

3.3. The effect of tasks number

If we want to show the impact of the tasks' number on the solutions. We should solve three instances from DS1 where the number of employees is 10 but we change the projects of the software. The three projects of the software have a various tasks'number: 10, 20, and 30. As clarify in Table 5, when the tasks increased, directly the cost and duration of the whole project increased too. We observed from the outcomes that when increasing the tasks number, the problem becomes harder. Figure 3 show the convergence curve of instances that taken from DS1 with different numbers of tasks.

Table 5. Results with (10, 20, 30) tasks number

Instance	Fitness	Cost	Duration
10_10_5 (employee-task-skill)	1.4086	799900	6.0875
10_20_5 (employee-task-skill)	4.1210	2134200	19.8680
10_30_5 (employee-task-skill)	7.0611	3141900	39.1913

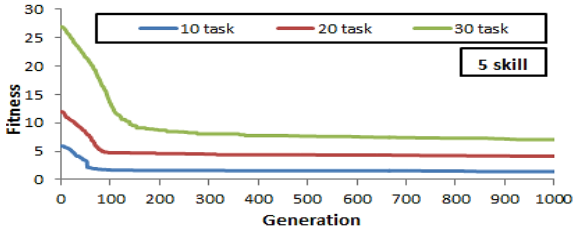

Figure 3. Convergence curve of different number of tasks for DS1

3.4. Comparison with other methods

In this section, we will focuse on comparing the outcomes of our experiments with another constructive metaheuristic. We compared our results with the results obtained in [6]. Table 6 shows that the GWO is better than MMAS-HC, ACO, and IWD. To assess the GWO algorithm quality, we compared its results with the results obtained in [13], the GWO algorithm exceeded the results of IWDSTD and IWDAS as shown in Table 7.

Table 6. Comparison the fitness of GWO with IWD, MMAS-HC and ACO

Instance	5 employee 10 task 5 skill	10 employee 10 task 5 skill	15 employee 10 task 5 skill	10 employee 20 task 5 skill
GWO	2.1191	1.4086	1.2911	4.1210
IWD	3.353	2.763	2.186	6.603
MMAS-HC	3.311	2.617	1.996	6.211
ACO	3.558	2.638	2.083	6.369

Table 7. Comparison the GWO with IWDSTD and IWDAS

Instance	5 employee 10 task 5 skill	5 employee 10 task 10 skill	10 employee 10 task 5 skill	10 employee 10 task 10 skill
GWO	2.1191	3.0275	1.4086	2.2432
IWDSTD	3.1073	3.6253	2.5629	2.8881
IWDAS	3.1446	3.6603	2.5313	2.8992

At last, we compared our results with the results of firefly algorithm in [16], Table 8 shows that GWO exceeded FA for all instances except one. We concluded from the values of Tables 6-8 that the GWO algorithm was better than the other algorithms because it found values of fitness function for most instances that used in this work less than the values of fitness function for other algorithms compared with them.

To show the effectiveness of GWO algorithm we performed a statistical test and computed the relevant p-values to compare the GWO algorithm with all algorithms mentioned above. Table 9 show wilcoxon signed ranks test results. According to the p-value computed by wicoxon signed ranks test [29] in the Table 9. It is evident that GWO algorithm has significant performance in all evaluated aspects. For the first five comparisons (GWO with the rest algorithms), we can say that the significant performance of GWO verifies at confidence interval 80%. The reason for the confidence interval is relatively low justifiable by the number of samples is less than six. Regarding the performance of GWO comparing with FA, it also has significant performance at higher confidence interval which is 95%.

Table 8. Comparison the GWO with FA

Instance	5 employee 10 task 5 skill	10 employee 10 task 5 skill	5 employee 10 task 10 skill	10 employee 10 task 10 skill	10 employee 20 task 5 skill	15 employee 20 task 5 skill	10 employee 30 task 5 skill	10 employee 30 task 10 skill
GWO	2.1191	1.4086	3.0275	2.2432	4.1210	3.8677	7.0611	9.5219
FA	3.4065	2.6158	4.2416	2.3206	6.3151	4.4841	9.6307	8.3977

Table 9. Wilcoxon signed ranks test results

Comparison	p-value
GWO versus IWD	0.125
GWO versus MMAS-HC	0.125
GWO versus ACO	0.125
GWO versus IWDSTD	0.125
GWO versus IWDAS	0.125
GWO versus FA	0.03906

4. CONCLUSION

In this paper, I would like to employ GWO algorithm in order to solve the SPSP problem. SPSP is a nessesary issue for the management of the software project and for the problem of NP-hard. We applied the GWO-SPSP on different instances from two datasets and studied the impact of variation of employees and tasks number. We conclude from a study of the variation of employees' number for the first dataset that has five skills, increasing employees' number leads to reduce the time of the project, but we could not find an explanation for the cost' values. As for the second dataset, where the number of skills is ten, increasing the number of employees leads to a reduction in project time and cost. We conclude from the study of increasing the number of tasks that the issue becomes complex and the algorithm needs more cycles as the number of tasks increases in order to find the best solution for scheduling, meaning that the time and cost increase. Numerical outcomes demonstrated that the proposed GWO algorithm outperforms IWD, ACO, MMAS-HC, IWDSTD, IWDAS and FA for all instances, except one instance that FA is outperform the GWO.

REFERENCES

[1] I. Sommerville, "Software engineering," *ninth edition, Addison-Wesley Publishing Company*, United States, ISBN-13: 978-0-13-703515-1, ISBN-10: 0-13-703515-2, pp. 792, 2010.

[2] M. Ahmad *et al.*, "Management Issues in Software Development," *Proceedings of the 9th WSEAS international conference on Software engineering, parallel and distributed systems*, 2010, pp. 232–237, doi: 10.1007/3-540-54194-2_28.

[3] M. Kaur, "Particle swarm optimization approach to software project scheduling," *International Journal of Academic Research and Development*, vol. 3, no. 1, pp. 308-310, 2018.

[4] M. Kaur, "Software project scheduling using ant colony optimization," *International Journal of Advanced Research and Development*, vol. 3, no. 1, pp. 214-217, 2018.
[5] A. C. Biju, T. A. A. Victoire, and K. Mohanasundaram, "Retracted: An Improved Differential Evolution Solution for Software Project Scheduling Problem," *Scientific World Journal*, vol. 2015, pp. 9, 2015, doi: 10.1155/2015/232193.
[6] B. Crawford *et al*., "Solving the Software Project Scheduling Problem Using Intelligent Water Drops," *Technical Gazette*, vol. 25, no. 2, pp. 350-357, 2018, doi: 10.17559/tv-20160629224348.
[7] E. Alba and J. F. Chicano, "Software project management with Gas," *Information Sciences*, vol. 177, no. 11, pp. 2380-2401, 2007, doi: 10.1016/j.ins.2006.12.020.
[8] F. Habibi, F. Barzinpour, and S. J. Sadjadi, "Resource-constrained project scheduling problem: review of past and recent developments," *Journal of Project Management*, vol. 3, no. 2, pp. 55-88, 2018, doi: 10.5267/j.jpm.2018.1.005.
[9] S. E. Almshhadany and L. M. Ibrahim, "Using Multi-objective Artificial Fish Swarm Algorithm to Solve the Software Project Scheduling Problem," *International Journal of Computer Applications*, vol. 181, no. 16, pp. 6-13, 2018, doi: 10.5120/ijca2018917753.
[10] M. Á. V. Velázquez's, A. G. Nájera, and H. Cervantes's, "A survey on the Software Project Scheduling Problem," *International Journal of Production Economics*, vol. 202, pp. 145-161, 2018, doi: 10.1016/j.ijpe.2018.04.020.
[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46-61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.
[12] T. R. Alreffaee and M. A. Alabajee, "Solving Software Project Scheduling Problem using Whale Optimization Algorithm," *IOP Conference Series: Materials Science and Engineerin*g, vol. 928, 2020, doi: 10.1088/1757-899x/928/3/032084.
[13] B. Crawford *et al*., "Self-configuring Intelligent Water Drops Algorithm for Software Project Scheduling Problem," *International Conference on Information Technology & Systems ICITS 2019: Information Technology and Systems*, 2019, pp 274-283, doi: 10.1007/978-3-030-11890-7_27.
[14] V. Rachman and M. A. Ma'sum, "Comparative analysis of ant colony extended and mix-min ant system in software project scheduling problem," *in Big Data and Information Security (IWBIS), 2017 International Workshop on. IEE*, 2017, pp. 85-91, doi: 10.1109/iwbis.2017.8275107.
[15] N. Nigar, "Model-based dynamic software project scheduling," *In Proceedings of 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 1042-1045, doi: 10.1145/3106237.3119879.
[16] B. Crawford, R. Soto, F. J. Parejas, C. Valencia, and F. Paredes, "Firefly Algorithm to Solve a Project Scheduling Problem," *Artificial Intelligence Perspective in Intelligent Systems Proceedings of the 5th Computer Science On-line Conference 2016 (CSOC2016)*, 2016, pp. 449-458, doi: 10.1007/978-3-319-33625-1_40.
[17] B. Crawford, R. Soto, F. Johnson, S. Misra, F. Paredes, and E. Olguín, "Software project scheduling using the hyper-cube ant colony optimization algorithm," *Tehnički vjesnik*, vol. 22, no. 5, pp. 1171-1178, 2015, doi: 10.17559/tv-20140519212813.
[18] F. Luna, D. L. González-Álvarez, F. Chicano, and M. A.Vega-Rodríguez, "The software project scheduling problem: A scalability analysis of multi-objective metaheuristics," *Applied Soft Computing*, vol. 15, pp. 136-148, 2014, doi: 10.1016/j.asoc.2013.10.015 .
[19] D. B. Hanchate and R. Bichkar, "Software Project Scheduling Management by Particle Swarm Optimization," *Oeconomics of Knowledge*, vol. 6, no. 4, pp. 24-54, 2014.
[20] B. Crawford, F. Johnson, R. Soto, E. Monfroy, and F. Paredes, "A Max-Min Ant System algorithm to solve the Software Projects Scheduling Problem," *Expert Systems with Applications*, vol. 41, no. 15, pp. 6634–6645, 2014, doi: 10.1016/j.eswa.2014.05.003.
[21] J. Xiao, X. T. Ao, and Y. Tang, "Solving software project scheduling problems with ant colony optimization," *Computers and Operations Research*, vol. 40, no. 1, pp. 33-46, 2013, doi: 10.1016/j.cor.2012.05.007.
[22] S. I. Khaleel and A. A. Al Thanoon," Design a Tool for Generating Test Cases using Swarm Intelligence," *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 10, no. 1, pp. 421-444, 2013, doi: 10.33899/csmj.2013.163468.
[23] H. Faris, I. Aljarah, M. A. Al-Betar, and S. Mirjalili, "Grey wolf optimizer: a review of recent variants and applications," *Neural Computing and Applications*, vol. 30, no. 2, pp. 413-435, 2017, doi: 10.1007/s00521-017-3272-5.
[24] H. Farughi, S. Mostafayi, and J. Arkat, "Healthcare Districting Optimization Using Gray Wolf Optimizer and Antlion Optimizer Algorithms (Case Study:South Khorasan Healthcare System in Iran)," *Journal of Optimization in Industrial Engineering*, vol. 12, no. 1, pp. 119-131, 2019.
[25] N. M. Hatta, A. M. Zain, R. Sallehuddin, S. Z. B. Abd. Rahim, and Y. Yusoff, "Recent studies on optimisation method of GreyWolf Optimiser (GWO): a review (2014–2017)," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2651-2683, 2018, doi: 10.1007/s10462-018-9634-2.
[26] M. A. Sen and M. Kalyoncu, "Grey Wolf Optimizer Based Tuning of a Hybrid LQR-PID Controller for Foot Trajectory Control of a Quadruped Robot," *Journal of Science*, vol. 32, no. 2, pp. 674-684, 2019.
[27] J. S. M. Alneamy and M. M. A. Dabdoob, "The Use of Original and Hybrid Grey Wolf Optimizer in Estimating the Parameters of Software Reliability Growth Models," *International Journal of Computer Applications*, vol. 167, no. 3, pp. 12-21, 2017, doi: 10.5120/ijca2017914201.
[28] Z. M. Gao and J. Zhao, "An Improved Grey Wolf Optimization Algorithm with Variable Weights," *hindawi, Computational Intelligence and Neuroscience*, vol. 2019, pp. 1-13, 2019, doi: 10.1155/2019/2981282.
[29] J. Derrac, S. García, D. Molina and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1 , no. 1, pp. 3-18, 2011, doi: 10.1016/j.swevo.2011.02.002.