HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTER THESIS

Mr.
Khan, Muhammad Usman

Optimization of Project Scheduling Problem Using Multi-Objective Bat-inspired Algorithm and comparison with other Nature Inspired Algorithms

2019

Faculty of **Applied Computer and Life Sciences**

MASTER THESIS

Optimization of Project Scheduling Problem Using Multi-Objective Bat-inspired Algorithm and comparison with other Nature Inspired Algorithms

Author:
Khan, Muhammad Usman

Study Programme:
Applied Mathematics in Digital Media

Seminar Group:
MA15w1-M

First Referee:
Prof.Dr.Thomas Villmann

Second Referee:
Dr.Tina Geweniger

Mittweida, March 2019

Acknowledgements

I would like to thank my supervisors Prof.Dr.Thomas Villmann and Dr.Tina Geweniger for there tremendous support and invaluable advice throughout my thesis. Prof.Dr.Thomas Villmann and Dr.Tina Geweniger excellent supervision made the research easier for me and there knowledge helped me to go through all the difficult phases of the thesis with no trouble.

I am very thankful to Prof.Dr.Peter Tittmann, the dean of Master's in Applied Mathematics in Digital Media for his assistance and expert opinions throughout the Master's program and also in thesis. I am thankful to all the professors that helped me improve my self both technically and socially, so that I could finally be able to successfully finish my studies.

I would like to thank all my friends, colleagues and classmates for their constant support during my studies. Finally, I would dedicate my thesis to my parents and wife who have always been a continuous support during my life and without there love, care and prayers this was not possible.

Abstract

In the practice of software engineering, project managers often face the problem of software project management. It is related to resource constrained project scheduling problem. In software project scheduling, main resources are considered to be the employees with some skill set and required amount of salary. The main purpose of software project scheduling is to assign tasks of a project to the available employees such that the total cost and duration of the project are minimized, while keeping in check that the constraints of software project scheduling are fulfilled. Software project scheduling (SPSP) has complex combined optimization issues and its search space increases exponentially when number of tasks and employees are increased, this makes software project scheduling problem (SPSP) a NP-Hard problem.

The goal of software project scheduling problem is to minimize total cost and duration of project which makes it multi-objective problem. Many algorithms are proposed up till now that claim to give near optimal results for NP-Hard problems, but only few are there that gives feasible set of solutions for software project scheduling problem, but still we want to get more efficient algorithm to get feasible and efficient results.

Nowadays, most of the problems are being solved by using nature inspired algorithms because these algorithms provide the behavior of exploration and exploitation. For solving software project scheduling (SPSP) some of these nature inspired algorithms have been used e.g. genetic algorithms, Ant Colony Optimization algorithm (ACO), Firefly etc. Nature inspired algorithms like particle swarm optimization, genetic algorithms and Ant Colony Optimization algorithm provides more promising result than naive and greedy algorithms. However there is always a quest and room for more improvement.

The main purpose of this research is to use bat algorithm to get efficient results and solutions for software project scheduling problem. In this work modified bat algorithm is implemented where a different approach of random walk is used. The contributions of this thesis are to: (1) To adapt and apply modified multi-objective bat algorithm for solving software project scheduling (SPSP) efficiently, (2) to adapt and apply other nature inspired algorithms like genetic algorithms for solving software project scheduling (SPSP) and (3) to compare and analyze the results obtained by applied nature inspired algorithms and provide the conclusion.

Keywords: Software Project Scheduling Problem, SPSP, Nature Inspired Algorithms, GA (Genetic Algorithms), ACO (Ant colony optimization), Bat algorithm, Modified multi-objective bat algorithm (MMOBA).

Contents

List of Figures

List of Tables

1 Introduction

In the practice of software engineering, project administrator frequently has to tackle the problem of software project management. A manager or administrator of a project has to initialize, schedule, begin, monitor and finish the project in project management. As it is a very tough job that's why there is need to optimize project scheduling. Software Project Scheduling Problem (SPSP) is an alteration of project scheduling problem (PSP). In it manager has to assign tasks to employees and should keep in view that total cost and total duration of project is minimized. Here main resources are considered to be the employees with some skill set and required amount of salary. In PSP most of the time there is only one objective while in software project scheduling problem (SPSP) there are mainly two objectives (1) minimize the cost of total project and (2) minimize the duration of total project, so this makes it a multi objective optimization problem. The main reason of software project scheduling is to assign tasks of project to the available employees efficiently, so that the total cost of project and the total duration of project completion are decreased while keeping in check that the constraints of software project scheduling are fulfilled.

Software project scheduling (SPSP) has complex combined optimization issues and its search space increases exponentially when number of tasks and employees are increased, this makes software project scheduling problem (SPSP) a NP-Hard problem. It is related to famous resource constrained project scheduling problem that is also referred as a NP-hard problem. Software project scheduling problem (SPSP)consists of a set of employees and a set of tasks and each task must have some employees working on it, who have the skills according to the skills required by that task to be done.

The goal of software project scheduling problem is to minimize total cost and duration of project which makes it multi-objective problem. Many algorithms are proposed up till now that claims to give optimal results for NP-Hard problems, but only few are there that gives feasible set of solutions for software project scheduling problem, but still we want to get more efficient algorithm to get feasible and efficient results.

Nowadays, most of the problems are being solved by using metaheuristics and nature inspired algorithms because these algorithms provide the behavior of exploration and exploitation [49]. Exploitation searches for the solution in a limited region where the optimal solution is found in hope of getting better solutions while exploration searches for the solution in huge space and tries to explore whole search space. For solving software project scheduling problem (SPSP) some of these nature inspired algorithms are used like genetic algorithms, Ant colony optimization algorithm (ACO), Firefly etc. Algorithms like genetic algorithms and ant colony optimization (ACO) provides more promising result than naive and greedy algorithms. However there is always a quest

and room for more improvement.

This thesis provides the overview of some of these nature inspired algorithms, review of most of the work done up till now on software project scheduling problem (SPSP) and solution of solving software project scheduling problem(SPSP) by using bat algorithm which is also a nature inspired algorithm.

1.1 Research Problem

Software project scheduling (SPSP) is a multi-objective optimization problem where administrator of the project has to assign set off employees to set of tasks, with some constraints that each task must have at least one employee working on it and the employees working on that task must have the skills required by a task to be done. It is variation of resource constrained project scheduling problem and is NP-hard.

1.2 Research Question

To find the optimal solutions for software project scheduling problem (SPSP), we will look into how nature inspired algorithms can be optimized and applied to software project scheduling problem (SPSP). In short, the goal of thesis is to answer the following research question:

How to use nature inspired algorithms to solve software project scheduling problem effectively?

To give answer to the above mentioned research question this thesis mainly focuses on exploring different nature inspired algorithms and metaheuristics that have been used to efficiently solve software project scheduling problem(SPSP). The reason for choosing nature inspired algorithm is because they have proved to be efficient most of the times in case of NP-Hard problems and because they possess the property of exploration and exploitation [49].

1.3 Contributions

The contribution of this thesis is to give detailed information of some nature inspired algorithm that how these algorithms can be used to solve software project scheduling problem.

The contributions are as follows:

1. To adapt and apply modified multi-objective bat algorithm (MMOBA) for solving software project scheduling (SPSP) efficiently.
2. To adapt and apply other already applied nature inspired algorithms like genetic algorithms for solving software project scheduling (SPSP).
3. To compare and analyze the results obtained by applied nature inspired algorithms and provide the conclusion.

1.4 Research Methodology

A systematic way to solve a problem is known as research methodology [21]. Mainly the research is of two types (1) rate Basic research (2) Applied research. Basic research is also known as underlying or conceptual research. In this one has to provide the deep insight and information about the problem, logical explanation of the problem and how to solve it and the conclusions about it. At the end basic research helps other researcher in their applied research. While applied research means to solve a problem or some specific problems with known algorithms or with new researcher defined algorithm. Applied research becomes helpful for others who are doing basic research.

This research thesis focuses on both basic and applied research. The answer to our research question "Solving software project scheduling problem (SPSP) using Bat algorithm" will provide basic research of software project scheduling problem (SPSP) and the algorithms that have been used to solve this problem, and as an applied research in this thesis bat algorithm will be applied to the software project scheduling problem (SPSP). As bat algorithm is nature inspired algorithm we will investigate software project scheduling problem (SPSP) with other nature inspired algorithms. Experiments will be conducted in our research on the benchmark dataset for software project scheduling problem (SPSP) and results will be compared and evaluated based on certain measures.

1.5 Research Strategy/Design

Both research types, basic and applied can further be of two types (1) Qualitative strategy and (2) Quantitative strategy. Qualitative focuses on the quality of results it is descriptive and it do not involve experimentation while on the other hand quantitative strategy is non descriptive, based on experiments, provides results with graphs and charts etc.

This thesis will conduct experimental strategy. The hypothesis concluded by doing qualitative research on software project scheduling problem (SPSP) in previous chapters that nature inspired algorithms provide efficient and viable results for software project scheduling problem (SPSP). As bat algorithm has already been proposed and is proven

efficient and viable in many cases so we want to apply bat algorithm on the important software project scheduling problem, to check its validity and viability.

2 Extended Background

In this chapter extended background of software project scheduling, resource constrained project scheduling and some famous nature inspired algorithm used for solving scheduling problem is presented.

2.1 Resource Constrained Project scheduling problem

Project scheduling is a mechanism where tasks are matched, or issued to data center resources. There isn't any scheduling algorithm that is absolutely perfect enough to solve task scheduling problem because of different scheduling objectives. A fine scheduler applies a suitable accommodation, or applies collaboration of different scheduling algorithms according to applications requirements [11]. Project scheduling is one of the problems that has received tremendous amount of attention and researchers have carried out multiple studies on this problem.

Time of solving a problem depends upon the algorithm that is applied to solve that problem. Time complexity of an algorithm is its execution time depending on the input. If a problem can be solved in polynomial time and it has a polynomial time algorithm than the problem is efficient enough to be executed or computed on any machine. In computational complexity theory, all problems are divided on the basis of certain resources into different complexity classes [11, 12].

There are 4 main complexity classes which are as following:

Polynomial Time: When the execution time of a computation, $m(n)$, is no more than a polynomial function of the problem size, n. More formally $m(n) = O(n^k)$ where k is a constant.

P: Those problems that can be solved in polynomial time belong to class P means it can be executed quickly in polynomial time.

NP: A problem is said to be in the NP class if it can be executed and solved in polynomial time by a non-deterministic Turing machine [13]. Problems of this kind take too much time to solve but can be quickly verified with some samples.

NP-Hard: A problem belongs to NP- Hard class if the algorithm for solving that NP-Hard problem can be transformed for solving another NP class problem. If a problem is NP-hard, no one should believe it can be solved in polynomial time. Resource constrained project scheduling problem is a combinatory optimization problem. It considers resources that have limited availability [50], tasks of different duration's and require-

ments are linked by precedence relations. The goal is to get a schedule of tasks and resources that gives minimum duration of project to get complete, keeping in view the availability of resources and precedence relations of tasks. It is considered to be a NP-Hard problem so it belongs to NP-Hard class.

2.2 Nature-inspired algorithms

2.2.1 Genetic algorithm

Every organism in this world has a mechanism by which they are built from small building blocks of life. These principles are encrypted in the genes of an entity, which further are attached together into lengthy strings type structure known as chromosomes [1]. Each gene is a certain quality of that entity, for example hair color, height etc. The characteristics that are mostly hidden in an organism are called genotype, for example inherited diseases, nature factor. While the physical characteristics of an organism that can be seen are called phenotype e.g. nose shape, eye color etc. Sharing of genes occur when two organisms mate with each other. The offspring generated may have traits of both parents. The process of mating is called recombination. Sometimes a gene is mutated in this process. It is called mutation. The algorithms that mimic the above nature process are called genetic algorithms. They depict a sharp exploitation of a random search which is used to solve optimization problems [2]. Genetic algorithms follow the principal of Charles Darwin theory "survival of the fittest."

The set of generated solutions of a problem is called a population. Fixed numbers of populations are specified at the start of the algorithm. The whole process is executed until the required number of generation is achieved. There are two methods to initialize a population. Initialize the population with random solutions i.e. random initialization, Initialize the population with known heuristic of the problem i.e. heuristic initialization.
Coding of chromosomes is represented as binary vectors. It is written as

$$X = Chromosomes \tag{2.1}$$

$$X = [x_1, x_2, ..., x_n] \tag{2.2}$$

And the number of vector elements are given by $l(X) = n$, where n is the length of the vector. And set $[x_1, x_2, ..., x_n]$ represents the genes. Gene is a single vector element or a connected part of chromosome. Fitness of each candidate solution in each population is calculated by a fitness function. Fitness function is made on the basis of objective function. It calculates the feasibility of solution according to the constraints and objectives of

the problem. The proportional fitness is given by

$$f_{prop}(x_i) = \frac{a \cdot b_i}{\sum_{j=1}^{N_c} b_j} \tag{2.3}$$

Where b_i is the evaluation of the genes, a is a constant and N_c is the number of individuals.
Let us take an example to explain our point. Let there be two chromosomes

$$X_1 = [01001] \tag{2.4}$$

$$X_2 = [10001] \tag{2.5}$$

The corresponding profit values are given by [5 3 2 7 1], the aim is to maximize the profit. First b_1 is calculated as $b_1 = 0 \cdot 5 + 1 \cdot 3 + 0 \cdot 2 + 0 \cdot 7 + 1 \cdot 1 = 4$, now we calculate the fitness for chromosome X_1 using equation 2.1 as

$$f(X_1) = \frac{4}{10} = .4 \tag{2.6}$$

Similarly b_2 is calculated as $b_2 = 1 \cdot 5 + 0 \cdot 3 + 0 \cdot 2 + 0 \cdot 7 + 1 \cdot 1 = 6$, now we calculate the fitness for chromosome X_1 using equation 2.1 as

$$f(X_2) = \frac{6}{10} = .6 \tag{2.7}$$

So clearly as we can see $f(X_2)$ is better.
The selection and marriage scheme in genetic algorithm is stated as that the probability of selection is proportional to the evaluation or fitness function i.e. everybody get a chance to reproduce.The most commonly used selection schemes are random selection, roulette selection scheme, linear ranking and random selection of fittest. The aim is the continuously increasing quality of the populations. This is achieved by a scheme called random selection. In which we generate two random numbers $r_1 \geq 1$ and $r_2 \leq N_c$. Second step is the selection of chromosomes based on the following

$$X_{sel} = \begin{cases} X_{r1}, & \text{if } f(X_1) > f(X_2) \\ X_{r2}, & \text{otherwise} \end{cases}$$

The advantage of this scheme is that not all $f(X_i)$ has to be calculated in each generational step.
Crossover in Genetic algorithms (GA) plays the role of operator to combine two parents and create a new offspring from their mixed traits. Two parents are selected according

to the fitness proportionate selection and crossover is applied among them. There are different techniques of crossover. The aim of crossover is faster search by overcoming local minima/maxima. A simple crossover technique is single point crossover. In this technique two parent solutions are selected both are divided into two parts and to create an offspring solution its first half is taken from the first parent and the second half is taken from the second parent. In mathematical terms it is represented as, we say generate a random number $r \leq 1$, then we split the given chromosomes at position r and finally we switch fragments to obtain children. In case of two point crossover we generate two random numbers for two chromosomesX_1 and X_2 as $1 \leq r_1 \leq l(X_1)$ and $1 \leq r_2 \leq l(X_2)$ respectively, and then we split and switch genes at r_1 and r_2. In uniform crossover we generate a random binary vector of length $l(X)$ and then we switch the genes/vector elements of the parent chromosomes if the random vector element is zero. The last and fourth crossover type is cut and splice crossover, in which we generate two random numbers for two chromosomes X_1 and X_2 as $1 \leq r_1 \leq l(X_1)$ and $1_2 \leq l(X_2)$ respectively, and then cut chromosome X_1 at r_1 and chromosome and X_2 at r_2 and then just splice to generate child's.
Mutation also plays an important role in adding diversity by changing the traits of chromosomes [3]. Chromosomes are mutated after some time with some defined probability so that the fitness of the solution can be improved but in some cases mutation can also worsen the fitness. Mutation rate should be low because it may destroy the information. Like crossover there are different techniques of mutation. A simplest one is bit string conversion. For example, let's say our solution is represented in a binary vector. Here a bit is selected by probability function and is flipped i.e. if it is 0 than it is converted to 1 and vice versa. In mathematical terms it is called equal distribution in which we generate a random number $1 \leq l(X)$ and then switch the gene at position r given as

$$x'_r = 1 - x_r \tag{2.8}$$

Genetic algorithms basic structure is as follows:

1. Choose coding for chromosomes
2. Initialize start population
3. Evaluate individuals of parent generation based on evaluation and fitness function
4. Selection of pairs of individuals according to a marriage
5. Produce children by recombination (Crossover)
6. Mutate children
7. Replace individuals of parent population by children according to replacement scheme.
8. Stop,otherwise continue with step 3

2.2.2 Particle swarm optimization

In 1995, Eberhart and Kennedy proposed Particle swarm optimization (PSO) computation technique [51, 52]. This technique was conceived by observing the social interaction between birds during flocking. In a flock every bird follows the path which is taken by the leader. After some time, some other bird from the flock replaces the leader bird and the flock starts following that new leader bird.This is valid for migrant birds flying in arrow shape.

Particle swarm optimization algorithm firstly initializes the population. In each iteration, position and velocity of the particles are changed according to the best local and global positions of other particles. The particles are than evaluated by a fitness function on its current position. The stopping criterion depends on the problem under consideration but typically it is the number of maximum iteration or the maximum fitness value achieved [51]. The principle of swarm intelligence states that it is a multi-agent system with self organization and signs of intelligence. There are certain rules to explain it. Every bird/object knows three rules, first one is cohesion which says to stay in the middle of surroundings, second states move away if someone gets too close and is called separation, third is alignment which states move in the same direction as your neighbours. Further for migrating birds the are two set of rules, first one states that to use boast or uplift of the bird infront of you and second is to select position to look straight ahead.
The aim of the particle swarm optimization is to basically minimize the cost function $f : R^m \rightarrow R$, then we find the particle or the possible candidate as

$$P_i : f(X_i) \leq f(X_j) \forall i, j \in [1, ..N] \tag{2.9}$$

$$P_i : i = arg_j min f(X_j) \tag{2.10}$$

Where N is the size of the swarm and X_i is the position of the particle P at i.

Algorithm:
The algorithm for particle swarm optimization is given by:

For all particles P_i for $i \in [1, ..N]$

- Initialize the population $X_i \in [X_{min}, X_{max}]$
- Set the best known position of P_i by self experience to be $b_i = X_i$
- Set the best known position of the swarm by social experience also known as

global best to be

$$g = \begin{cases} b_i, & \text{if } i = 1 \\ b_i, & \text{if } f(b_i) < f(g) \\ g, & \text{otherwise} \end{cases}$$

- Initialize velocity

$$v_i \in [-|X_{max} - X_{min}|, |X_{max} - X_{min}|] \tag{2.11}$$

While not finished, for all P

- Generate randomly r_b and $r_g \in [0,1]$
- Update velocity

$$v_i^{new} = \alpha v_i + \beta r_b (b_i - X_i) + \gamma r_g (g - X_i) \tag{2.12}$$

- Update particle position

$$x_i^{new} = x_i + v_i^{new} \tag{2.13}$$

- Update best position. If $f(X_i) < f(b_i)$, update $b_i = X_i$. If $f(b_i) < f(g)$, update $g = b_i$
- g is the best solution.

Where α, β and γ represents the switching directions of particles, X_{max}, X_{min} represent the upper and lower limit of search space. The inertia allows the particle to move in the same direction with same velocity. Personal influence improves the individuals and cause the particles to return to a previous better position, whereas social influence makes particles to follow the best neighbour position. If the velocity is low then the algorithm is slow and if the velocity is high then algorithm becomes unstable, so there is no guarantee to reach the global optimum. To resolve this issue the concept of sub swarms is considered in which we separate g_k for each group.
To interpret the algorithm there are two term used, one is exploration and the other one is exploitation. In exploration we search the border region at search space which includes diversification (inertia), whereas in exploitation we conduct locally oriented search to get close to an optimum solution and it includes intensification(personal and social influence). So α, β and γ are chosen in such a way that exploration versus exploitation is balanced.

2.2.3 Ant Colony Optimization (ACO)

The basic Ant colony optimization (ACO) algorithm was initiated by Marco Dorigoin early 1990s [6, 7]. These algorithms were motivated by the etiquette of ants. Ants are social insects [8] and live in colonies. The inspiration of making the Ant colony optimization (ACO) algorithm was the action of ants to get the shortest path to find food. At first, ants explore the area or space around their nest for searching the food. They leave

a chemical named pheromone while moving on the ground. When an ant locates the food it gets the food and returns to the nest from the same way by which it has come from, in this way pheromone on that path becomes double. Ants can smell pheromone. When choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food. The pheromone trails will guide other ants to the food source. The indirect communication between the ants via pheromone trailsâĂŤknown as stigmergy enables them to find shortest paths between their nest and food sources. Ant colony optimization (ACO) algorithm copies the method of ants in which they locate the shortest path amid food and nest. Often a graph like search space is made for a problem and Ant colony optimization (ACO) is applied on that search space i.e. graph.

Ant moves in a graph node to node. The probability that it will move to j node from i node is calculated by following probability function

$$p_{ij} = \frac{(T_{ij}^{\alpha})(\eta_{ij}^{\beta})}{\sum(T_{ij}^{\alpha})(\eta_{ij}^{\beta})} \tag{2.14}$$

- T_{ij} is the amount of pheromone on the edge i, j
- α is a variable to control the impact of T_{ij}
- η_{ij} is the heuristic, depending on the problem
- β is a variable to control the impact of η_{ij}

Pheromone update can be done globally and locally. Global pheromone update is done at the end of iteration. The best solution is picked up and pheromone is updated on the edges where best solution was located. Local pheromone update is done during the traversal of ants among the graph nodes. Pheromone is updated according to following equation

$$T_{ij} = (1-p)T_{ij} + p\Delta T_{ij} \tag{2.15}$$

- If only i, j belongs to best tour
- p is the evaporation rate of pheromone.
- ΔT_{ij} is the participated amount of pheromone

Algorithm:

1. Set parameters ,initialize pheromone trails
2. While not finished do,

- Construct ant solution
- Apply local search to improve solution and is optional.

- Update pheromones

Construction of Ant solution:

1. Walk on construction graph $G_c(V,E)$, which is a fully connected construction graph
2. Construct solution based on finite set of available solution components C
3. Start with empty partial solution set $s_k^0 = \emptyset$
4. With each construction step, s_k^p is extended by adding solution components from $N(s^p)$
5. Choice of solution component C_{ij} from $N(s^p)$, guided by stochastic mechanism biased by the pheromone level associated with each element C_{ij}

Update Pheromone:

1. Decrease pheromone level also known as evaporation
2. Increase pheromone level associated with paths describing good solutions.

Where $C = C_{ij}$ in which C_{ij} is solution component associated with edge (i, j), s_k is solution associated with ant k, s_k^p is the partial solution, $N(s_k^p)$ which is a subset of C is the set of possible solution components taking restrictions into account. The aim was basically to find solution $s* : f(s*) \leq f(s^k)$, $\forall\ k$, $k \in [1, ..m]$, where m is the number of ants.

2.2.4 Firefly Algorithm

Fireflies are distinguished by their blazing light produced by biochemical process bioluminescence [14, 15]. This blazing light acts as the main function for catching the prey and mating between the fireflies. The lanterns are the organs in fireflies that produce bioluminescence. Male firefly's unique flashing pattern attracts female firefly. The light becomes dense by the increase of distance. Firefly algorithm mimics the process of mating and hunting of fireflies. It was developed by Xin-She Yang at Cambridge University in 2007. Some rules in firefly algorithm are as follows

- All fireflies should be unisex [16]
- When the brightness is decreased attractiveness is also decreased so the both should be proportional to each other
- Dazzle of firefly is calculated by the objective function [16]

Algorithm:

- Objective function $f(x), x = (x_1, ..., x_d)$
- Generate initial population of fireflies $x_i (i = 1, 2, ..., n)$
- Light intensity I_i at x_i is determined by $f(x_i)$
- Define light absorption coefficient
- while $(t < MaxGeneration)$
  - for $i = 1 : n$ all n fireflies
    - for $j = 1 : i$ all n fireflies
    - if $(I_j > I_i)$, Move firefly i towards j in d-dimension; end if Attractiveness varies with distance r via $exp[r]$ .Evaluate new solutions and update light intensity.
    - end for j

- end for i
- Rank the fireflies and find the current best
- End while
- Post process results and visualization

The light intensity is given by the following equation if we take into account the absorption coefficient and it varies with the square of distance r

$$I = I_0 e^{-\gamma r^2} \tag{2.16}$$

The population of fireflies is initialized by the following equation

$$x_{t+1} = x_t + \beta_0 e^{-\gamma r^2} + \alpha\varepsilon \tag{2.17}$$

where β_0 is the attractiveness at $r = 0$.The third term is randomization with α being the randomization parameter, and ε is a vector of random numbers drawn from a Gaussian distribution or uniform distribution at some time.If $\beta_0 = 0$, it becomes a simple random walk. On the other hand, if $\gamma = 0$, it reduces to a variant of particle swarm optimization. The attraction is represented by the second term. The attractiveness of fireflies is given by the following relation

$$\beta = \beta_0 e^{-\gamma r^2} \tag{2.18}$$

If we now say that firefly j is the brightest firefly and firefly i is attracted towards it, then this movement from i to j is given by

$$x_i = x_i + \beta_0 e^{-\gamma r_{i,j}} (xj - xi) + \alpha\varepsilon \tag{2.19}$$

2.2.5 Cuckoo Search

Cuckoos are the birds that lay their eggs into nests of other birds, they remove other bird's eggs from the nest to increase the hatching probability of their own eggs [18]. When the owner birds see that the eggs in the nests are not theirs, they either remove the eggs of cuckoo from the nest or flew away to make another nest somewhere else. So some cuckoo's has the ability that they can mimic their eggs in color and pattern according to the host specie. This reduces the probability of their eggs being abandoned and thus increases their productivity [17]. When the cuckoo's egg is hatched the first thing the chick do is that it kicks out other eggs from the nest so its chance of getting food is increased.

Cuckoo search algorithm is motivated by the above mentioned actions of cuckoos. It uses following rules.

- One egg is laid in randomly chosen nest by one cuckoo at a time

- Those nests which have high quality of eggs (solutions) will be forwarded into next generation
- Total number of accessible nests is predefined and the probability that a cuckoo picks a nest is calculated by $pa \in [0, 1]$

3 Literature Review

3.1 Genetic Algorithms

Genetic algorithm is a replica of biological evolution used in artificial systems. Genetic algorithm can be used to solve software project scheduling problem. This iterative approach searches for the best solution in a huge search space. It first take the population of randomly selected chromosomes, each chromosome is evaluated by a fitness function. Each chromosome is represented by a vector and each cell is its genes. The genetic operators that play an important role in making the new generation are crossover, reproduction and mutation. The chromosomes are encoded according to the choice of person as it is flexible. Crossover takes two chromosomes from previous generation and swaps some bits from each other to create a new child. Mutation modifies genes by the probability given by a user. Genetic algorithms (GA) use large search space in which the whole population moves to find the best solutions. Some of the work done on genetic algorithms (GA) to solve software project scheduling problem (SPSP) is presented below.

T. Hanne, S. Nickel [25] uses MOAEA/D(Multiobjective Evolutionary Algorithm with Decomposition) algorithm to solve software project scheduling problem (SPSP) in 2003. MOAEA/D is a decomposition algorithm that decomposes the solution into sub problems and solves each sub problem and each problem is assigned some weights. In this paper author develop a discrete event simulation model, some of the variables are fixed for task assignment and scheduling in this model. This software project scheduling problem (SPSP) is a multi-objective optimization problem solved using the mentioned model. The population which is developed by MOEAD is distributed as a Pareto front giving the best solutions for software project scheduling problem.

Enrique Alba and J. Francisco Chicano[26] uses genetic algorithms's on software project scheduling problem (SPSP) in 2007. The software project scheduling problem (SPSP) is encoded into binary chromosome. The solution to software project scheduling problem (SPSP) is first changed into a employees task size matrix, in which each value of a cell is dedication of an i^{th} employee to that j^{th} task. This X matrix is non negative matrix. It is expected that no employee should work overtime so the maximum dedication of employee is considered to being in range of $[0, 1]$. Here 8 values of dedication are considered to be encoded by three bits string.

They take cosecutive three bits string check there dedication degree and fill in the matrix. Each bit string is a solution here and are called chromosomes. Its fitness and fesibility

is calculated by following function.

$$f(x) = \begin{cases} \frac{1}{q}, & \text{if the solution is feasible} \\ \frac{1}{(q+p)}, & \text{otherwise} \end{cases}$$

Where,

$$q = w_{cost}.p_{cost} + w_{dur}.p_{dur} \tag{3.1}$$

And,

$$p = w_{penal} + w_{undt}.undt + w_{reqsk}.reqsk + w_{over}.p_{over} \tag{3.2}$$

Where p is the penalty for infeasible solutions, that is, for those solutions that violate some of the problem constraints. Whereas, w_{cost} and w_{dur} and are weights associated with the cost and the duration, respectively. Where w_{penal} is a penalty that is imposed when the assignments between tasks and employees are incomplete, p_{over} is the amount of overtime work performed by the employees in a project, $undt$ is the monetary cost of a project, $reqsk$ is the time required to finish a project, and w_{over}, w_{undt}, and w_{reqsk} are weights that are associated with the overload, cost, and time objectives, respectively.

D.Sundar and colleagues presented MOGA(Multi-Objective Genetic Algorithm) algorithm to solve software project scheduling problem (SPSP) in 2010[27]. First initial population is generated, initial population is the divided into m subpopulations according to number of objectives, in software project scheduling problem (SPSP) there are two objectives. Now for each subpopulation fitness of these objectives is calculated. Best chromosome is selected and two parent chromosome are selected from population. Crossover is applied between parents. Get the best offspring O from the off springs populations. After that apply crossover between the best chromosome X and best offspring O, If new generated offspring is better than replace $X$with $O2$. Iterate for all chromosomes and until best pareto is obtained. The results shown by this algorithms are considered to be good. The prioritization of constraints involved in this problem is also done by the MOGA, and this shows effective results over manual allocation techniques.

J. Francisco Chicano again presented GA's for software project scheduling problem (SPSP) in 2011 [28]. Many multi-objective metaheuristics are used to solve the problem here. They have been compared with 39 publicly available instances that cover all type of scenarios large, medium and small. The algorithm used in this paper and all the results will assist project managers in the difficult tasks of project scheduling problem. To measure the performance of the algorithm used in this paper the quality of the Pareto set is maintained. Two things are used here hyper volume(HV)indicator and the attainment surfaces. They have chosen a set of parameters for a good comparison among each algorithm used in this paper NSGAII(Non-dominated Sorting Genetic Algorithm II), SPEA2(Strength-based Evolutionary Algorithm) and MOCell(Multiobjective Cellular Genetic Algorithm) also GDE3(Generalized Differential Evolution). Population size of 100

is maintained archive size is also 100 for algorithms like PAES, MOCell and GDE3. The algorithm stops at 100000^{th} iteration. The results are analyzed in this paper in two ways. HV hyper volume indicator and attainment surfaces to find the quality of multi objective algorithm for software project scheduling problem. This tells that PEAS has the best value of Pareto front, MOCell performed good on hyper volume. Main algorithms used in this paper NSGAII and SPEA2 have the worst HV values. The attainment surface allows differentiating between the regions of whole objective space which is explored in multi objective algorithms. PAES has outperformed all the algorithms with low cost and log duration. Whereas, NSGAII and SPEA2 have given the results with high project cost and less duration.

Constantinos Stylianou and colleagues in 2011 [29] presented approach of solving SPSP and team staffing using Genetic algorithms (GA) to construct project optimal schedule and assign tasks to most expertized employees. Genetic algorithms (GA) used in this paper uses different objective functions to handle the constraints and results gathered proved to be optimal when objective functions were stand alone however when they were combined, Genetic algorithms (GA) did not give optimal solutions.

Milena Karova in 2012 presented [30] in this paper that Genetic Algorithm technique can be used to search for near-optimum solution, minimum duration and cost of project. "The given model is named as IGAPM (Implementation Genetic Algorithm for Project Management).The solutions of algorithm are chromosome matrix where the rows are employees and columns are set of activities or tasks. The chromosome is represented in binary format. (1) A value of 0 when the employee is unable to perform the task because of its skills. (2) Row normalization of the chromosome. The performance of IGAPM is represented as steady-state. For given number initiate with a constant configuration of genetic operators and variables, fitness function's different values are relatively constant, IGAPM implementation provides a way to solve this problem by efficient use of genetic algorithms."

Leandro L. Minku and colleagues in 2012 presented [31] a theoretical analysis of performance of genetic algorithms for software project scheduling problem (SPSP) with some improvements in the design of genetic algorithms like normalization of employee's dedications, enhanced mutation operator and fitness functions that are focused in providing feasible solutions. They compared genetic algorithms with normalized and without normalized dedications and stated that normalization helps in getting high hit rate and good quality solutions

Abel Garcia-Najera and colleagues in 2014 proposed [32] software project scheduling problem (SPSP) model that represents operations of software companies better. It considers skills of employees into 4 levels, beginner, junior, senior, expert for assigning employees to tasks. Same objectives of software project scheduling problem (SPSP) are followed in addition with reducing overtime also as an objective. They used simple

Genetic algorithms (GA), however it slightly differs in mating selection technique as the first parent is chosen by fitness proportion method while the second is chosen by the average similarity of population. This paper claims to have better results by using their proposed approach than NSGA-II.

Leandro L. Minku in 2014 gave [33] - analysis of different variations of genetic algorithms. As many of the variations of evolutionary algorithm have been applied to software project scheduling problem, yet there are many whose performance is dependent on the design choices for evolutionary algorithms. In this paper author presented the run time complexity for project scheduling problem by looking into the performance of evolutionary algorithm and project scheduling problem in general. Authors made an improved evolutionary algorithm (EA) design that merges normalization for the dedication of employees for all tasks to make sure that no employee works overtime, the fitness function is generated that requires few parameters and provide a good path guiding towards optimal solutions, and more improved design for crossover and mutation. Normalization removes the extra amount of time for calculating the overworks of employees and allows evolutionary algorithm to just focus on the solution quality. It gives opportunity in finding balance between the dedications of employees for all tasks and allows employees to use the workload in better whenever a task is started or finished. They derived a general upper bound and lower bound for the expected overtime for big classes of evolutionary algorithm. The glitch in this paper is that despite using the normalization technique some solutions still find a way to escape from the local optima. While in general it gives the best results and normalization is a key component for the efficiency of this algorithm presented in this paper.

Mayowa Ayodele in 2015 presented [34] the work in which software project scheduling problem (SPSP) is split into two sub problems and different algorithms are applied to each sub problem. In this paper Genetic algorithms (GA) and EDA (Estimation of Distribution Algorithm) are two methods that are used separately in each sub problem. Genetic algorithms (GA) are used to improve the quality of tasks and employees ordering while EDA focuses on how to generate mode assignment.

Constantinos Stylianou and colleague also presented the analysis of different genetic algorithms in 2016 [35].

3.2 Ant Colony Optimization

Jing Xiao and her colleagues proposed to solve software project scheduling problem (SPSP) using ant colony optimization algorithm in 2012 [36]. In this paper they made a construction graph for each task which consists of dedications as rows and employees as columns, ants traverse through each task graph, returning back is not allowed. This paper presents six heuristics for selecting a node; they have different aspects of trade-

offs for software project scheduling problem (SPSP). They compared the results with Genetic algorithms (GA) and proved that their proposed work gives better solutions with better hit rate.

Wei-Neng Chen and her colleagues in 2013 proposed [37] method with an event-based scheduler (EBS) and an ACO algorithm. "The proposed method shows a scheme by a task list and a schemed employee allocation matrix. According to this method both difficulties of scheduling a task and scheduling employee to that task can be managed. In the event-based scheduler (EBS), starting time, release time of resources from completed tasks and joining and leaving of an employee in project are used as events. Main idea of this approach is to adjust employee's allocation to changed and unchanged events. It tackles the conflict issue of tasks preemption and provides flexible approach in resources allocation. For solving planning issue, an ACO algorithm is further designed. Experimental results on 83 instances prove that the given method is very efficient. The main things of this algorithm are divided into two parts: event-based scheduler (EBS) and ACO for tackling planning problems. Experimentation shows that this scheme is good and the given algorithm provides better plan that gives minimum cost and duration."

Broderick Crawford and colleagues in 2013 presented [38] ACO hypercube approach to solve software project scheduling problem (SPSP). Just an overview of ant colony optimization (ACO) is given and no experimentations and results are provided.

Bharti suri in 2013 also [39] provided brief information, experimentation and results of solving software project scheduling problem (SPSP) using ACO algorithm.

Olfa Dridi in 2013 proposed [40] a hybrid ant colony optimization (HACO) approach to solve software project scheduling problem (SPSP). They presented a bi-directional ant colony optimization (ACO) methodology in which 2 variation of ants is used to find better solution. The first type of ant is from ACS-Cost colony. It optimizes the costs of the tasks and the whole project. Second type of ant is from ACS-time colony it optimizes the time of total project. Authors claim in this work that their hybrid bi-directional ant colony optimization (ACO) works better and provide better results than simple ant colony optimization (ACO) algorithm.

Broderick Crawford in 2014 proposed a min-max ant system to solve software project scheduling problem (SPSP) [41].Min-max ant miner approach proposed by (StÃijtzle and Hoos, 2000) is applied on software project scheduling problem (SPSP). They claim that their proposed solution provide better results. And in 2015 author combined its work of solving software project scheduling problem (SPSP) with ant colony optimization (ACO) hyper cube [38] and min- max ant colony optimization [41], they proposed a hybrid algorithm by combining these two [42] and claimed to get viable solutions.

Jing Xiao, in 2015 presented [43] combination of ant colony optimization and MOEAD evolutionary algorithm to solve software scheduling problem. Two heuristics are examined in this model to search better in software project scheduling problem (SPSP). Experiments are conducted on publicly available 39 instances that cover all type of things large medium and small. The results are then compared with other algorithm NSGAII. This model did not outperformed on NSGAII for complex instances. The procedure of the model is as follows: Input is taken = number of ants, instances of software project scheduling problem, no of groups, neighbors size and other basic parameters, Output = pareto front, Initialization, Generate the initial values of ants evaluate each ant as the process of MOEAD, generate new population, update according to the fitness, output the solution.

3.3 Firefly Algorithm

Broderick Crawford, Ricardo Soto, Franklin Johnson in 2016 [48] solved software project scheduling problem (SPSP) using nature inspired firefly algorithm. Here each firefly is displayed to be a unique solution and by using main features of this algorithm, the problem is solved as follows. First each task is sorted according to the task precedence graph (TPG), than parameters are initialized like size of population, light intensity etc. Than fireflies are created and are represented in the form of a matrix than all constraints are validated and at the end fitness or light intensity of a firefly or a solution are calculated and the location of firefly is updated. This paper claims that this algorithm on software project scheduling problem (SPSP) gives good solutions on smaller instances only.

3.4 Metaheuristics

A. C. Biju in 2015 [44] proposed a Differential evolution strategy or method to solve software project scheduling problem (SPSP). Differential evolution method has proved out to be very reliable algorithm to solve NP hard problems. As it has solved RCPSP successfully so it is also applied to software project scheduling problem (SPSP) problem in this paper. They proposed a differential evolution (DE) algorithm with different mutation operators for software project scheduling problem (SPSP) and proposed IDE-SPSP. It can be showed as a graph based search problem for which differential evolution (DE) is used. The new operators give good base to use the domain based heuristics for improving the performance of IDE-SPSP algorithm. All the information for software project scheduling problem (SPSP) is stored in an output file including task precedence graph (TPG) etc. Model is build according to instances and the division operation is applied on different tasks and then at the end proposed IDE method is applied to get best solutions. The population has to go through following operations. Mutation: For each vector a mutant vector is generated. Crossover: it is introduced to increase diversity. This paper

claims that IDE-SPSP produces most reasonable solutions.

Broderick Crawford, Ricardo Soto in 2014 presented [45, 47]software project scheduling problem (SPSP) as a combinatory optimization problem and provides the metaheuristics to solve software project scheduling problem (SPSP) as they have proven to be capable of solving difficult problems. The metaheuristics that they discussed in this paper are genetic algorithm (GA), Ant colony, Simulated Annealing, Variable Neighborhood Search, and Memetic Algorithms. In this paper they provide the overview of metaheuristics to solve software project scheduling problem (SPSP) and provide the reason that why metaheuristics provides better results than other methods. They did not provide the comparative results for these provided algorithms.

Xiaoxia Huang in 2015 [46] proposed a mean variance model for project scheduling. For solving complex problems they use lower and higher semi variance as a hybrid algorithm that integrates with genetic algorithms. The flow of the whole algorithm is as follows: first LSV and HSV are calculated with inverse uncertainty distribution method to solve the problem for the general cases. Then the solutions vectors are represented according to Genetic algorithms (GA) vectors. Then comes the initialization phase in which some constraints etc. are defined, the selection is done by roulette wheel spinning after that crossover and mutation operators are applied and then the solution is implemented in cellular automata and this hybrid algorithm procedure continues until the required numbers of generations are processed. The paper claims that the proposed algorithm provides efficient results and it can help the managers in software project scheduling problem (SPSP) in profitable way. Sometimes the parameters are to be provided via experts due to lack of history so their approach provides a better way to schedule projects in an optimal way.

4 Research Methodology and Proposed Solution

4.1 Project scheduling problem (PSP)

Project scheduling problem (PSP) is one of the most well-known problem in software industry that manager face for managing software projects. It is related to resource constrained project scheduling problem. Alba and Chicano in 2007 first focused on this problem and provide its solution using genetic algorithms [26]. The main resources that are involved in software project scheduling problem (SPSP) are tasks, employees and skills. Set of tasks are the project requirement that need to be done, set of employees are those who will work on different tasks and set of the skills are those skills that are needed to complete the whole project. Each task needs a set of skills to be done and each employee has its own set of skills. Therefore it should be handled carefully that the set of employees that are doing a task should have the skills needed by that task. Overall goal of software project scheduling problem (SPSP) is to assign employees to tasks in a way that the cost and duration of whole project is minimized.

For understanding of software project scheduling problem (SPSP) we will first describe the resources in detail and then the mathematical formulation to represent the software project scheduling problem (SPSP) model. Resources required by software project scheduling problem and the model of software project scheduling problem are defined below:

4.1.1 Skills

Skills are the set of abilities that an employee possesses. Every task requires some skills in order to be done. An employee can have all the skills or some of required skills for example, Graphics expert, UML expert, Automation expert, testing and programming expertise etc. The set of skill of whole project is union of the skills of all tasks required to complete that project. It is defined as $SK = \{sk_1, sk_2, sk_3, ..., sk_{|n|}\}$ where$|n|$ is the total number of skills required to complete a project.

4.1.2 Tasks

Tasks are the important and necessary work or activities that are required by software project in order to get accomplished. These activities can be like analyzing, designing, testing, coding etc. The set of tasks that are necessary to complete a project is

defined as $T = \{tsk_1, tsk_2, tsk_3, ..., tsk_{|t|}\}$, where |t| is the total number of tasks in a project. There is different precedence among the tasks, in order to complete a project each task must be done in a sequence keeping in view its precedence. Commonly the precedence among tasks is represented in a graph structure known as task precedence graph (TPG). It is a directed non-cyclic graph that contains a set of vertices and edges and is represented as $G(V,E)$. Each vertex in a graph represents a task, so the set of vertices is equal to the set of tasks. An edge between two vertices represents the precedence among corresponding tasks in vertices. And all the edges in TPG make a set of edges E. We can say that when an edge $(t_i, t_j) \in E$ than task $i(t_i)$ must be done before task $j(t_j)$.

Each task contains two features (1) a set of skills and (2) effort.

- Set of skills required by a task j can be represented as $t_j^s = \{t_j^{s_1}, t_j^{s_2}, ..., t_j^{s_{|k|}}\}$ where $|k|$ is the required number of skills for task j.
- Effort is the real number and shows the workload amount of task j. It can be represented as e i.e. $t_j^e = 10$.

4.1.3 Employees

Employees are the most important resource in software project scheduling problem. Each employee possesses various skills. Employees are allocated to the tasks and it is the work of project manager to assign employees to right tasks. The main problem is to create a schedule of employees and tasks. The set of employees can be represented by $Emp = \{e_1, e_2, ..., e_{|E|}\}$ where $|E|$ is the total number of employees that are there to complete the project. Each employee contains three features (1) a set of skills, (2) salary and (3) maximum dedication.

- Set of skills of employee i can be represented as $e_i^s = \{e_i^{s_1}, e_i^{s_2}, .., e_i^{s_{|g|}}\}$ where $|g|$ is the number of skill possessed by employee i.
- Salary is the real number and defines the monthly amount to be paid to an employee i. It can be represented as sal i.e. $e_i^{sal} = 56,030$.
- Maximum dedication of an employee is the maximum amount of work that an employee gives to a project. It can be represented as $maxd$ i.e. e_i^{maxd}. $e_i^{maxd} \in [0,1]$, if employee i maximum dedication $e_i^{maxd} = 1$ than employee works completely on that project if $e_i^{maxd} < 1$ than it may be possible that employee is doing part time job.

4.1.4 Model

The necessary elements representation and description of software project scheduling problem model is represented in the following Table 4.1.

Following in this section is an example of the software project scheduling problem

Element	Description
$SK = \{sk_1, sk_2, sk_3, ..., sk_{\|n\|}\}$	Set of skills required by a project
$T = \{tsk_1, tsk_2, tsk_3, ..., tsk_{\|t\|}\}$	Set of tasks required by a project
$G(V,E)$	Task precedence graph (TPG)
$V = \{t_1, t_2, t_3, ..., t_{\|T\|}\}$	Set of vertices containing all tasks
$E = \{(t_i, t_j), ..., (t_n, t_{\|T\|})\}$	Set of edges i.e. task i must be done before task i
t_j^s	Set of skills required by a task to be done
t_j^e	Effort of a task
$Emp = \{e_1, e_2, , e_{\|E\|}\})$	Set of employees working on a project
e_i^s	Set of skills possessed by an employee
e_j^{sal}	Employee monthly salary
e_i^{maxd}	Employee maximum dedication toward project
$M = (m_{ij})$	Employee $i's$ dedication towards task j
t^{start}	Starting time of a task
t^{finish}	Finishing time of a task
t^{cost}	Total cost of a task
t^{dur}	Total duration of a task
p^{cost}	Overall projects cost
p^{dur}	Overall projects duration
e^{overw}	Employee overwork if total dedication exceeds $maxd$
p^{overw}	Overwork of a project, sum of all employees overwork

Table 4.1: **Elements representation for SPSP**

model.

Example:

Number of employees $= 3$
$E = \{e_1, e_2, e_3\}$

Number of tasks $= 5$
$T = \{t_1, t_2, t_3, t_4, t_5\}$

Number of skills $= 4$
$SK = \{sk_1, sk_2, sk_3, sk_4\}$

Information of employees is as follows

$e_1^s = \{e_s^1, e_s^3, e_s^4\}$,

$e_1^{sal} = 51240$, $e_1^{maxd} = 1$

e_2^s ={e_s^1, e_s^2, e_s^4},

$e_2^{sal} = 65121, e_1^{maxd} = 1$

e_3^s ={e_s^2, e_s^3, e_s^4},

$e_3^{sal} = 54522, e_1^{maxd} = 1$, The following Figure.1 represents task precedence graph (TPG)

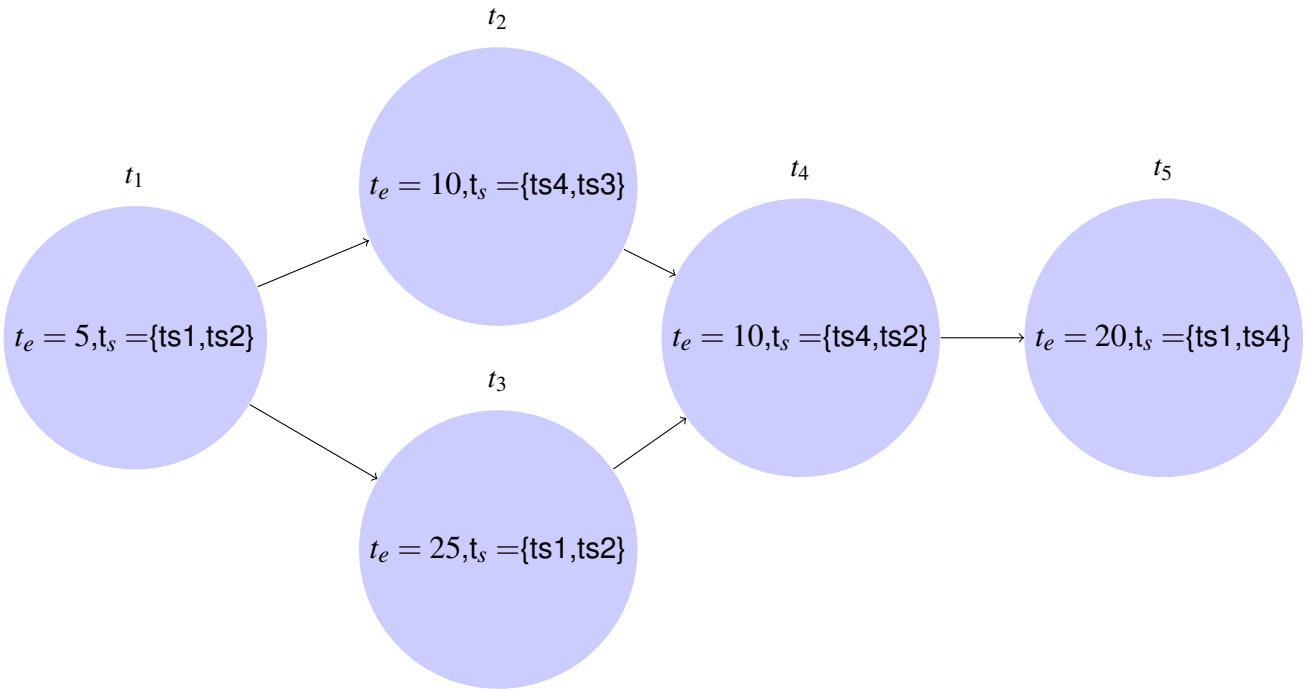

Figure 4.1: **Figure** 1

Each task has number of skills that is a subset of Sk and an effort.

A solution of software project scheduling problem can be represented by the matrix M of size $E * T$. The cell in a matrix is represented as m_{ij} and shows the dedication of the employee i for the task j. It can be in between $[0, maxd]$. If employee i dedication to task j is 1 than it means it is completely dedicated to that task. If its dedication is 0.5 than it means it spends half of its time on that task. If it is 0 than it means that he is not doing that task.
The solutions of software project scheduling problem may not be feasible sometimes. There are some constraints to check its feasibility. First we have to make sure that at least one employee is assigned to each task i.e. there shouldn't be any task in which no

employee is assigned. It can be described as following

$$\sum_{i=1}^{E} m_{ij} > 0, \forall j \in 1,2,3,...T \tag{4.1}$$

Second constraint is that the union of skills of those employees that are assigned to task j should be equal or super set of the set of skills of task j. It can be described as follows.

$$t_j^{skills} \subseteq \bigcup_{i|m_{ij}} > 0, \forall j \in 1,2,3,...T \tag{4.2}$$

The solution matrix should looks like following table, when all dedications are assigned Two constraints described above are easy to find in the solution matrix as shown in

M_{ij}	T_1	T_2	T_3	T_4	T_5	T_6
E_1	1.00	0.50	0.00	0.25	0.00	0.00
E_2	0.25	0.50	1.00	0.50	0.00	0.25
E_3	0.50	1.00	0.00	0.50	1.00	0.25

Table 4.2: **Sample solution matrix**

table 4.2. In software project scheduling problem the quality of solution is dependent on three factors

1. Feasibility of solution
2. Total duration of project
3. Total cost of project according to the matrix solution

Feasibility can be calculated by checking the constraint's requirements. Now we have to calculate the total cost and total duration of whole project. These can be done as follows.
First we have to calculate the starting time, finish time and duration for each task of project.
Duration of task j can be calculated by following equation.

$$t_j^{duration} = \frac{t_j^{effort}}{\sum_{i=1}^{E} m_{ij}} \tag{4.3}$$

For example according to table 2, the duration of task 4 can be calculated as following

$$t_4^{duration} = \frac{10}{0.25 + 0.50 + 0.50} = 8 \tag{4.4}$$

Now we will calculate the start time and finish time of task j. It is calculated with respect to task duration and its precedence relation described in task precedence graph (TPG). Those task's starting time that are not dependent on any other tasks can simply be 0

because there is no dependency of any other task, while those task's starting time which are dependent on other tasks is the maximum ending time of those tasks on which a task is dependent. Each task starting and finish time can be calculated easily from task precedence graph (TPG) because it is acyclic. It can be calculated by following equations.

$$t_j^{start} = \begin{cases} 0, & \text{if } \forall k \neq j(t_k, t_j) \in A \\ max[t_k^{end} | (t_k, t_j) \in A, & \text{otherwise} \end{cases}$$

$$t_j^{end} = t_j^{start} + t_j^{dur} \tag{4.5}$$

As we have calculated starting time, finish time and duration of each task. Now we can calculate total duration p^{dur} of whole project very easily. The ending time of the last task in task precedence graph (TPG) will be the total duration of the project and it is formulated as follows

$$p^{dur} = max[t_j^{end} | \forall k \neq j(t_j, t_k) \notin A \tag{4.6}$$

After that we have to calculate total cost p^{cost} of the project. For this first we have to calculate the cost of each task and it can be done by following equation.

$$t_j^{cost} = \sum_{i=1}^{E} e_i^{salary} . m_{ij} . t_j^{dur} \tag{4.7}$$

Now we can calculate the total cost of the project by the following equation

$$p_{cost} = \sum_{j=1}^{T} t_j^{cost} \tag{4.8}$$

In addition to all of this we can have another constraint or another objective that is over time work of the project, it means the sum of overtime of all employee's over work. The overtime can increase mistake chances in employee's work and in turn it may cause the delay in project duration as it will require more time to clear those mistakes or errors. So it is useful that it should be kept in focus what is the total overtime of a project in order to keep the track of project quality. To calculate the overtime work of an employee, its dedication on every task and those tasks starting and finishing time is required. The following equation calculates workload of an employee on instance t.

$$e_i^{work}(t) = \sum_{j | t_j^{start} \leq t \leq t_j^{end}} m_{ij} \tag{4.9}$$

An employee is doing overwork if its workload is exceeding his/her maximum dedication.Overwork of an employee can be calculated by following equations.

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

$$e_i^{over} = \int_{t=0}^{p_{dur}} ramp(e_i^{work}(t) - e_i^{maxded})dt \tag{4.10}$$

Now the sum of all employees overwork is the total overwork of the project p^{overw}. And can be calculated as follows.

$$p^{overw} = \sum_{i=1}^{E} e_i^{over} \tag{4.11}$$

A feasible project is the one which have no overwork duration i.e. $p^{overw} = 0$.

4.2 Algorithm used other then Nature Inspired Algorithm for the problem:

Algorithms exist that can find an optimal solution (one which produces the minimum/maximum value of the objective function) in a reasonable time (relative to the size of the problem) for the very simplest schedule-related problems in particular, for simple versions of workforce scheduling and basic project scheduling. One of the approaches that is used commonly is called

Construction Algorithms (including Greedy Algorithms), which start with an empty or incomplete solution (e.g. where no tasks are scheduled and/or no resources are assigned), and incrementally make it more complete (e.g. by scheduling one additional task and/or assigning one additional resource at a time).

Advantages:

- The simplicity.Greedy algorithms are generally easier to write as well as explain. All you need to do is check your neighbors and move to the larger one until you have found the end.
- It is efficient. Here we should not confuse efficiency with accuracy. Greedy algorithms may not always be the most accurate, but they are generally very efficient, as you only observe local possible moves.

Disadvantages:

- Cons are obvious. It does not not always find the most optimal path. Because you always give the priority to what has more weightage without knowing the bigger picture.
- Greedy algorithm doesn't typically refine its solution based on new information.

4.3 Data collection and Data Analysis

Data sets are collected through research, experimentation and through questionnaire in quantitative research. In our research the dataset used for the software project scheduling problem will be collected from "Problem/Instance generator" [22]. Dataset generated from this problem/Instance generator has been used in many other related works on SPSP.

Problem/Instance generator is a java program that takes simple parameters for SPSP and automatically generates the dataset for SPSP. The parameters to be set are the no of tasks, no of employees, etc.

Main components of a data set of SPSP are employees, skills, tasks, employee's salary, task's effort, and task precedence graph (TPG). The parameters have to be set in a configuration file. Sample configuration files are provided here [22]. For example:

- **Int-gen-20-15-t6-7.conf:** it generates data file with 15 employees that are uniformly distributed among 20 tasks with 6 or 7 skills per task

4.4 Local Search

Local Search is an iterative procedure that moves from one solution in S to the next (until some stopping criterion is satisfied).Let us consider a simplest example to start with, imagine a climber who is ascending a mountain on a foggy day 1. She can view the slope of the terrain close to her, but she cannot see where the top of the mountain is. Hence, her decisions about the way to go must rely only upon the slope information. The climber has to choose a strategy to cope with this situation, and a reasonable idea is, for example, choosing to go uphill at every step until she reaches a peak. However, because of the fog, she will never be sure whether the peak she has reached is the real summit of the mountain, or just a mid-level crest.
We need to define three concepts namely search space, the neighborhood relation, and the cost function in order to apply local search algorithms to a specific search or optimization problem.A computational problem upon which these three entities are defined is called a local search problem.A given computational problem P can give rise to different local search problems for different definitions of these entities.

4.4.1 Search Space:

Given a computational problem P , we associate to each instance of $y \in I_P$ a search space $S_P(y)$, with the following properties

- Each element $s \in S_P(y)$ represents an element $y \in S$.
- For optimization problems: at least one optimal element of $sk_p(y)$ is represented in $S_P(y)$. Together with an objective function $f : S(y)R$ that evaluates each feasible solution. We then seek given an instance ya feasible solution $x \in S(y)$ with minimum objective function value.

we have a valid representation or valid formulation of the problem if the previous requirements have been achieved. We write $S_P(x)$ as S for simplicity and refer to the elements of S as solutions.

4.4.2 Neighborhood Relation:

Def: Given a problem P , an instance $y \in I_P$ and a search space S for it, we assign to each element s$\in S$ a set $N(s) \subseteq S$ of neighboring solutions of s. The set $N(s)$ is called the neighborhood of s and each member $s' \in N(s)$ is called a neighbor of s.
For each s the set $N(s)$ needs not to be listed explicitly. In general it is implicitly defined by referring to a set of possible moves, which define transitions between solutions. Moves are usually defined in an intentional fashion, as local modifications of some part of s. The "locality" of moves (under a correspondingly appropriate definition of distance between solutions) is one of the key ingredients of local search, and actually it has also given the name to the whole search paradigm. Nevertheless, from the definition above there is no implication for the existence of "closeness" among neighbors, and actually complex neighborhood definitions can be used as well.
A neighborhood structure N may be represented by a directed graph $G = (V,A)$ where

- $V = S$
- $(u,v) \in A \iff v \in N(u)$

This graph is called the neighborhood graph. In general, it is not possible to store this graph completely. S usually has exponential size with respect to the instance.In order to avoid problems with size of the neighborhood graph, a neighorhood is usually described by operators:

- Let $F : S \to S$ be a function,
- For each feasible $s \in S, F(s)$ is a subset consisting only of feasible solutions, we

call F thus an allowed modification For every $s \in S$, we can define a neighborhood structure for a set AM of allowed modifications as follows

$$N(s) := F(s)|F \in AM \tag{4.12}$$

Suppose the neighborhood graph $G = (V,A)$ is connected, then for every (starting) solution $s \in S$, there exists a directed path to every other solution in S .In particular, we can provide a sequence of operations to s that result in an optimal solution $s \in S$.But we only need the later condition. A neighborhood N is called OPT-connected if, from each solution $s \in S$, an optimal solution can be reached by a finite sequence $s, s_1, ..., s_k, s_{k+1}$ of solutions $s_i \in S$ s.t. $s_{i+1} \in N(s_i)$ for $i = 1, ..., k, and s_{k+1}$ optimal.

4.4.3 Cost Function:

The selection of the move to perform at each step of the search is based on the cost function. The cost function F associates to each element $s \in S$ a value $F(s)$ that assesses the quality of the solution. For the sake of simplicity, we assume that the value of F is always a positive and integer, or in other words, that the co domain of F is the set of natural numbers.
For optimization problems, the cost function F must also take into account the objective function of the problem. Therefore the cost function is typically defined as a weighted sum of the value of the objective function and the distance to feasibility (which accounts for the constraints). Normally, the highest weight is assigned to the constraints, so as to give preference to feasibility over optimality.

4.4.4 Local Search Algorithm:

A local search algorithm, use the moved related to the definition of neighborhood we defined earlier to iterate by starting from an initial solution $s_{init} \in S$„ such that it navigates the search space. At each step it makes a transition between one solution s to one of its neighbors s'. When the algorithm makes the transition from s to s', we say that the corresponding move t_m has been accepted, and we also write that s' is equal to $s \oplus tm$. The selection of moves is based on the cost function, which as explained above accounts for the number of violated constraints, and, for optimization problems, also on the objective function of the problem.

Algorithm and Pseudo Code:

- procedure Local Search(Search Space S, Neighborhood N, Cost Function F);
- begin
- $s_0 := InitialSolution(S)$;
- $I_i := 0$;
- $while(StopCriterion(s_i, i))do$
- begin
- m:= $SelectMove(s_i, F, N)$;
- $if(AcceptableMove(m, si, F))$
- then $s_{i+1} := s_i \cdot m$;
- else $s_{i+1} := s_i$;
- $i := i + 1$

    * end
  - end

Local Search Techniques:
There are three basic Local search techniques

- Hill Climbing
- Simulated Annealing
- Tabu Search

Hill Climbing relies on the basic idea chosen by the climber in the metaphor proposed above: at each step of the search a move that "leads uphill" is performed. Hill Climbing is then improved by Simulated Annealing and Tabu Search. Simulated Annealing is based on probabilistic, memoryless decisions, whereas Tabu Search is based on the use of memory of previously visited solutions.

Hill Climbing:
Hill Climbing is a technique to solve certain optimization problems. In this technique, we start with a sub-optimal solution and the solution is improved repeatedly until some condition is maximized.The term comes from the goal of maximizing a certain function via an iterative improvement scheme of selecting at each step a move that improves the value of the objective function or reduce the distance to feasibility.We now list a set of common hill climbing techniques:

- The most well-known form of hill climbing is the so-called steepest hill climbing (SHC) technique. At each iteration SHC selects, from the whole neighborhood $N(s)$ of the current solution s, the element $s' = s \oplus t_m$ which has the minimum value of the cost function F . The procedure accepts the move t_m only is it is an improving move. Consequently, it stops as soon as it reaches a local minimum.
- Another popular hill climbing technique is random hill climbing (RHC). This technique selects at random (with or without a uniform probability distribution) one

element $s' = s \oplus t_m$ of the set $N(s)$ of neighbors of the current solution s. The move t_m is accepted, and thus s' becomes the current solution for the next iteration if t_m improves or let equal the cost function, otherwise s remains the current one for the next iteration.

Their stop criterion is based on the number of iterations without improving the value of the best solution.

Simulated Annealing:
The idea of this technique is to avoid cycling by randomization, i.e. simulate the annealing process from physics.The method got that name after an analogy with a simulated controlled cooling of a collection of hot vibrating atoms.We choose a solution$s' \in S$ randomly and then we accept solution only with a certain probability.
In the $i-th$ iteration, s' is accepted with probability,

$$min[1, e^{\frac{f(s')f(s)}{t_i}}] \tag{4.13}$$

where (t_i) is a sequence of positive control values with $\lim_{i\to\infty} t_i$. Often, (t_i) is defined in analogy to physics as

$$t_{i+1} := \alpha t_i, 0 < \alpha < 1 \tag{4.14}$$

Other variant of Simulated Annealing is given by Threshold Acceptance, where acceptance rule for $s' \in N$ is accepted if difference $f(s)f(s')$ is within some limit l_i. l_i also decreases with number of iterations, where $best := f(s)$.

Tabu Search:
Another deterministic strategy to avoid cycling is to store all visited solutions in a so-called tabu-list T. Tabu Search (TS) is a method in which keeping memory of features of previously visited solutions has a fundamental role. The basic mechanism of TS is quite simple and is given by

- a neighbor is only accepted if it is not contained in T
- At each iteration a subset $B \subseteq N(s)$ of the neighborhood of the current solution s is explored.
- The member of B that gives the minimum value of the cost function becomes the new current solution independently of the fact that its value is better or worse than the value in s.

To prevent cycling, there is a so-called tabu list, which is the list of moves which it is forbidden to execute. The tabu list comprises the last m moves, where m is a parameter of the method, and it is run as a queue; that is, whenever a new move is accepted as the new current solution, the oldest one is discarded. Notice that moves, not solutions, are asserted to be tabu. Therefore, a move t_m can be tabu even if when applied to the

current solution s it leads to an non visited solution. In other words, the basic TS scheme avoids visiting not only previous solutions, but also solutions having features presented in solutions already visited.For this reason, there is also a mechanism that overrides the tabu status of a move: If in a solution s a move t_m gives a large improvement of the cost function, then its tabu status is dropped, and the solution $s \oplus t_m$ is accepted as the new current one. The disadvantage is that also new solutions may be declared tabu, so the aspiration criteria is accept solution even if they are tabu, for exmaple based on objective function value.More precisely, this mechanism makes use of an aspiration function A. For each value t of the cost function, A computes the cost value that the algorithm aspires to reach starting from t. Given a current solution s, the cost function F, and the best neighbor solution $s' \in B, if F(s') \leq A(F(s))$,then s' becomes the new current solution, even if the move m that leads to s' has a tabu status.

4.5 Modified Multi-objective Bat Algorithm (MMOBA)

Bat algorithm is a nature inspired algorithm first proposed by Xin-She Yang in 2010 [23]. Bats are the only mammals that have wings and have this amazing capability of catching prey even in complete darkness due to echolocation. There are different species of bat and different size ranges. The capability of echolocation varies in different species. Micro bats uses echolocation extensively than other species of bats. The bats generates a loud impulse of sound and waits for the carefully observe and listens to the echo that bounces back from its surrounding objects. Bats have the ability that they can vary their pulse depending on the strategy of hunt. A bat algorithm was inspired by these amazing properties of bats.
Each bat in the algorithm represents a solution and has a velocity and position. It is updated in iterations according to following equations

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{4.15}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x) f_i \tag{4.16}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{4.17}$$

Where,

- f_{max} represents maximum frequency.
- β is a random variable whose value ranges between $[0, 1]$.
- v_i is velocity of i^{th} bat.
- x_i is the i^{th} position of bat.
- x represents solution of bat which has best fitness value obtained after comparison among all the n bats so far.

New solutions are generated through a random search in search space. Once solutions

are generated the best solution is selected. Now the new solutions or bats are generated from the current best solution by following equation.

$$x_{new} = x_{old} + \varepsilon A^t \tag{4.18}$$

- A^t is the average loudness of bats in a particular step
- ε is random value between $[-1,1]$ and represents direction and intensity of random walk.

As bats approaches near there prey usually the pulse emission rates increases and loudness usually decreases, so both these factors must be updated at each iteration step. They are updated by the following equations

$$A_i^{t+1} = \alpha A_i^t \tag{4.19}$$

$$r_i^{t+1} = r_i^0[1 - e^{(-\gamma t)}] \tag{4.20}$$

Here γ and α are constants, whose values are $\gamma > 0$ and $0 < \alpha < 1$.

Algorithm and Pseudo Code:

- Objective function $f(x), x = (x_1, ..., x_d)^T$
- Initialize the bat population $x_i (i = 1, 2, ..., n)$ and v_i
- Define the pulse frequency f_i at x_i
- Initialize pulse rates r_i and loudness A_i
- While ($t <$ Max number of iterations)

    1. Generate new solutions by adjusting frequency
    2. Update velocities and position or location using equation 4.13 and 4.14

        – if $(rand > r_i)$

            a) Selection a solution among the best solutions
            b) Generate a local solution around the selected best solution

        ∗ end if

            1. Generate a new solution by flying randomly

        ∗ if $(rand < A_i and f(x_i) < f(x))$

            1. Accept the new solutions
            2. Increase r_i and reduce A_i

        ∗ end if

1. Rank the bats and find the current best x

- end while

1. Post process results and visualizations

Modifications:

Bat algorithm was proposed in 2010 and up till now it is used in many application including Combinatorial Optimization and Scheduling, Inverse Problems and Parameter Estimation, data mining, image segmentation etc. [24] Like many other nature inspired meta heuristics algorithms, bat algorithm is simple, flexible and easy to implement.
According to the algorithm there is a problem balance between exploration and exploitation. In algorithm the possibility that $rand > r_i$ is very low to be ensured by the bat. Exploitation capability is dominating at the start of iterations whereas exploration dominates as iterations increase.So, optimization algorithm has to drive forward exploration capability at the first iterations then exploitation capability at the later iterations so as to reach optimum point.
The process of updating velocity and position increases exploration capability of bat algorithm whereas the generation process of candidate solution around the best solution increases exploitation capability. So this means algorithm will be good at exploration but bad at exploitation if the new candidates solutions are generated by equations $4.15 - 4.17$, algorithm will be bad at exploration but good at exploitation,if new candidate solutions are generated around best solution is according to equation 4.18, hence the balance is not preserved and algorithm can easily get trapped local minimum. So the solution to this issue is actually the pulse rate which provides the balance between exploration and exploitation. As the number of iterations increase the pulse rate also increases with it. The possibility of $rand > r_i$ decreases as the iteration proceeds, so this would mean that exploitation occurs at the first steps of iteration and exploration occurs at the preceding steps.
Loudness A decreases during the iterations. Accordingly the possibility of $A_i > rand$ is higher at the beginning of iterations but lower at the following iterations. So this would mean that the inclusion possibility of new candidate solutions into the bat population,which generated by exploration at the end of the iterations is weak. If the algorithm gets trapped into local minimum at the beginning of iterations, newly generated solutions also accumulate around such local minimum. Due to this reason, the elusion possibility of algorithm from local region decreases.
The bat algorithm has poor exploration ability but it can be improved by equalizing the loudness A and pulse emission rate r to the problem dimension. Because these are the two factors that effect all the dimensions of the solutions. So we basically assign these factors to each dimension in the solution separately so that both exploration and exploitation capabilities can be performed collectively. In modified bat algorithm each dimension j of solution i, which provides the $rand_j > r_{ij}$, approaches around the di-

mension j of the best solution and the rest dimensions of solution i keep on seeking the search space as compared to BA in which all the solutions satisfying $rand > ri$ converges to the best solution with entire dimension . It is given by following equation

$$x_{ij}^{t+1} = \begin{cases} x_{uj}^{t} + \varepsilon A_j^t, & \text{if } rand_j > r_{ij} \\ x_{ij}^{t}, & \text{otherwise} \end{cases}$$

- A_j^t represents the average loudness of dimension j of all the solutions at time t.
- u is the solution selected among best solutions.

Similarly for the dimension j of solution i where $rand_j > r_{ij}$, the loudness and pulse rate emission are updated as

$$A_{ij}^{t+1} = \begin{cases} \alpha A_{ij}^{t}, & \text{if } rand_j > r_{ij} \\ A_{ij}^{t}, & \text{otherwise} \end{cases}$$

$$r_{ij}^{t+1} = \begin{cases} r_i^0(1-e^{-\gamma t}), & \text{if } rand_j > r_{ij} \\ r_{ij}^{t}, & \text{otherwise} \end{cases}$$

Now according to these equations the dimensions j of solution i where $rand_j > r_{ij}$ are expected to search the space through exploration capability as iteration proceeds which previously exploit.Therefore the possibility of $rand_j > r_{ij}$ is reduced by increasing of pulse emission rate. Similarly apart from the dimensions above the other dimensions are expected to upgrade current solutions by exploitation capability at the following iterations which previously explore. So that $rand_j > r_{ij}$ possibility is retained for other dimensions at preceding iterations.Search range of best solution is narrowed for dimensions j of solution i by reducing loudness A that belongs to these dimensions.

In this work bat algorithm is modified to get better results, for random walk we defined a parameter Ω , which decides how many new solutions will be generated. These new solutions will be generated in way of random local search. The pseudo code for the modified bat algorithm is as follows

Algorithm and Pseudo Code:

Input: $f_{max}, f_{min}, \beta, a_{min}, a_{max}, \alpha, \gamma, \Omega, n, noOfIterations.$
Output: Non-Dominated solutions.

- Initialize the bat population $x_i, (i = 1, 2...n)$
- Initialize initial pulse, frequency and loudness for each bat.

- While (t < noOfIterations)
  - For each bat x
    1. Generate new solution by adjusting velocity and frequency according to respective equations mentioned above
  - if (rand > pulse rate of x)
    1. Generate Ω number of solutions using random local search.
    2. Select a solution among the best solutions.
  - end if
    1. Generate a new solution by flying randomly
  - if (rand< loudness of x $f(x) < f(x^*)$)
    1. Accept the new solutions
    2. Increase pulse and reduce loudness using equation mentioned above
  - end if
    1. Rank the bats and find the current best x^*
- end while
  1. Apply non-dominated sort and return non-dominated solutions.

4.6 Why Bat algorithm?

BAT uses frequency tuning and echolocation to solve the problems. Frequency variations help to mimic the true functionality of the problem. This feature is also available in other algorithms as well like particle swarm optimization (PSO) so bat algorithm (BA) has the advantage of particle swarm optimization (PSO). It uses automatic zooming and it is a prominent advantage on other metaheuristics algorithms. It has the ability that it can zoom into the area of search space where feasible and best solutions are found. It uses dynamic parameter control and in this way it can switch from exploration to exploitation in search space. Also theoretical analysis of BA by Huang et al. (2013) suggested that "bat algorithm (BA) has guaranteed global convergence properties under the right condition, and bat algorithm (BA) can also solve large-scale problems effectively" [24].

4.7 Evaluation measure

Optimal Pareto front is generated by taking 10 runs of each algorithm on software project scheduling problem (SPSP). To analyze and evaluate bat algorithm viability from other algorithms we will have following benchmark parameters for quality estimation of multi-objective algorithms.

- **Hyper-volume:** The hyper-volume calculates the volume in the objective space, which is covered by the solutions of the obtained Pareto front approximation.The hypervolume measure has been dominantly used for performance evaluation of nature inspired algorithms.The hypervolume gives the multidimensional volume of the portion of the objective space that is weakly dominated by an approximation set.Using the hypervolume of the dominated portion of the objective space as a measure for the quality of Pareto set approximations has received more and more attention in recent years. The reason is that this measure has two important advantages over other set measures. $1)$ It is sensitive to any type of improvements, i.e., whenever an approximation set A dominates another approximation set B, then the measure yields a strictly better quality value for the former than for the latter set algorithms for many years.$2)$As a result from the first property, the hypervolume measure guarantees that any approximation set A that achieves the maximally possible quality value for a particular problem contains all Pareto-optimal objective vectors.

$$HV(A) = \int_{z^{min}}^{z^{max}} \alpha_A(z)dz \tag{4.21}$$

$$\alpha_A(z) = \begin{cases} 1, & \text{if } \exists a \in A \text{ such that } z \succ a \\ 0, & \text{otherwise} \end{cases}$$

In practice, only the upper-bound vector $z^{max} \in \mathbb{R}^d$ is required to compute the hypervolume; this parameter is called reference point.

- **Spread:** This indicator measures the extent of spread by the set of computed solutions.
In order to compare the different algorithm we used the spreading metric in Equation

$$\Delta = \frac{\sum^{|F|} d_e + \sum_i^{|Q|} |d_i - \mu|}{\sum^{|F|} d_e + |Q|\mu} \tag{4.22}$$

$|F|$ is the number of objective functions, d_e is the distance between every extreme in the Pareto set we are measuring and the corresponding extreme in a Reference Pareto set, $|Q|$ is the population size, d_i is the sum of normalized Euclidean distance from point i^{th} to their neighbors in every objective function (point $(i+1)^{th}$ sorting every objective function), and μ is the mean of the distances of the Pareto

set we are measuring.

- **Generational Distance:** It measures how far are the elements in the computed approximation from those in the optimal Pareto front. The general distance (GD) indicates how close the obtained Pareto Fronts are to the true pareto front in multiobjective optimization problems.Given a candidate set $A = \{a_1, ..., a_N\}$ (in image space) and a Pareto front $F(P_Q)$=$\{y_1, ..., y_M\}$, the Generational Distance (GD)) are defined as follows:

$$GD(A) = \frac{1}{N}(\sum_{i=1}^{N} d_i^p)^{\frac{1}{p}} \tag{4.23}$$

where d_i denotes the minimal Euclidean Distance from a_i to $F(P_Q)$.

- **Inverted Generational Distance:** It measures the distances between each solution composing the optimal Pareto front and the computed approximation.Given a candidate set $A = \{a_1, ..., a_M\}$ (in image space) and a Pareto front $F(P_Q)$=$\{y_1, ..., y_M\}$, the Generational Distance (GD)) are defined as follows:

$$IGD(A) = \frac{1}{M}(\sum_{i=1}^{M} \bar{d}_i^p)^{\frac{1}{p}} \tag{4.24}$$

where d_i denotes the minimal Euclidean Distance from a_i to $F(P_Q)$.The main advantages of the IGD measure are twofold. One is its computational efficiency: The IGD measure can be easily calculated even for many-objective problems. The other is its generality: The IGD measure usually shows the overall quality of an obtained solution set A (i.e., its convergence to the Pareto front and its diversity over the Pareto front). Thanks to these nice properties, recently the IGD measure has been frequently used to evaluate the performance of optimization algorithms on many objective problems in the literature.

4.8 Proposed Method

The proposed method for solving software project scheduling problem (SPSP) is the modified version of multi-objective bat algorithm. According to theoretical research bat algorithm provides viable results in different problems so we assume that in software project scheduling problem (SPSP), it will also provide better results than the rest of the proposed algorithms for software project scheduling problem (SPSP). 6 datasets of different sizes will be generated and results of these datasets will be gathered on different algorithms. These results will then be compared by the results of modified multi-objective bat algorithm (MMOBA).

5 Experiments and Results

In thesis, software project scheduling problem using genetic algorithms results are reproduced.[28]The data sets used in this paper are generated using instance generator whose details are presented in the next section. Each data set is executed on different evolutionary algorithms 10 times. For comparison we have used MOEA framework that consists of multiple evolutionary algorithms. From these following algorithms are selected to execute software project scheduling problem and gathered results: DBEA(Improved Decomposition-Based Evolutionary Algorithm), MOEA- D (Multiobjective Evolutionary Algorithm with Decomposition), NSGAII(Non-dominated Sorting Genetic Algorithm II), OMOPSO(Multiobjective Particle Swarm Optimization), PESA2 (Pareto Envelope-based Selection Algorithm), Random Search, RVEA(Reference Vector Guided Evolutionary Algorithm), SPEA2(Strength-based Evolutionary Algorithm), VEGA(Vector Evaluated Genetic Algorithm), $\varepsilon-$MOEA(ε-Dominance-based Evolutionary Algorithm), GDE3 (Generalized Differential Evolution), PAES(Pareto Archived Evolution Strategy) and SMPSO(Speed-Constrained Multiobjective Particle Swarm Optimization) [53]. GDE3, MOEA-D, OMOPSO and SMPSO use real valued operators so for this we converted binary vector of chromosome into real number vector of dedications. Rest of the algorithms can be of any operator i.e. binary, real etc. Graph with calculated real Pareto fronts on each datasets is shown in which cost is on x-axis and duration is on y-axis and indicators like hyper-volume, spread, generational and inverted generational distance are calculated. Table 5.1 represents algorithms parameters and their values used in these experimentation.

Algorithm	Iterations	PopulationSize	pm.rate	bisections	pm.distributionIndex
DBEA	100000	100			
MOEAD	100000	100	1/L		
NSGAII	100000	100	1/L		
OMOPSO	100000	100			
PESA2	100000	100		4	
Random	100000	100			
RVEA	100000	100	1/L		
SPEA2	100000	100	1/L		
VEGA	100000	100	1/L		
e-MOEA	100000	100			
GDE3	100000	100			
SMPSO	100000	100	1/L	8	
PAES	100000	100	1/L	8	20

Table 5.1: **Algorithms with parameters values**

Algorithm	archiveSize	sbx.rate	mutationProbability
DBEA			
MOEAD		0.9	
NSGAII		0.9	
OMOPSO	100		1/L
PESA2			
Random			
RVEA		0.9	
SPEA2		0.9	
VEGA		0.9	
e-MOEA			
GDE3			
SMPSO	100		
PAES			

Table 5.2: **Algorithms with parameters values**

Specifications	MMOBA-Values
noOfIterations	1000
noOfBats (population size)	100
f_{min}	0
f_{max}	1
β	0.1
ω	20
A_{min}	0
A_{max}	1
α	0.99
γ	0.9

Table 5.3: **MMOBA with parameters values**

5.1 Data Set

The paper uses 6 data sets with following parameters

DataSets	No of employees	No of tasks	No of skills
Data Set1: i30-15g5	15	30	5
Data Set1: i10-5g5	5	10	5
Data Set1: i10-5p7	5	10	6-7
Data Set1: i20-15g10	15	20	10
Data Set1: i30-5g5	5	30	5
Data Set1: i40-20g10	20	40	10

Table 5.4: **Data sets with parameters**

5.2 Results

Hyper-Volume(Mean)						
	I10-5g5	I10-5p7	I20-15g10	I30-5g5	I30-15g5	I40-20g10
MMOBA	0.693938614	0.594480526	**0.667253201**	**0.465046955**	**0.611817457**	**0.446209688**
DBEA	**0.706986049**	**0.674420572**	0.4309389	0.19673789	0.426136805	NAN
MOEAD	0.243045019	0.204871916	0.311281035	1.08802E-15	0.129959366	0.142025684
NSGAII	0.130819567	0.077025762	0.151992537	0.170840062	0.082746746	0.06715621
OMOPSO	0.350558368	0.150850724	0.09369913	0.009131608	0.046166235	0.177894681
PESA2	0.172728381	0.16866737	0.075664801	0.183137562	0.048111157	2.58682E-15
Random	3.07532E-15	1.07692E-15	0.074641313	0.052740584	0.034560515	0
RVEA	0.356324809	0.094190035	0.121461589	0.047565473	0.177958441	0.050745783
SPEA2	0.297187686	0.190125122	0.16530752	0.10228894	0.123985751	2.10942E-16
VEGA	0.060130483	0.008331356	0.000483986	0.261880427	0.036613345	0.034252346
e-MOEA	0.678800543	0.63950606	0.267855872	0.128380458	0.129680624	NAN
GDE3	0.236470691	0.125907017	0.353945713	NAN	4.44089E-17	1.03251E-15
SMPSO	0.00177317	0.202266268	0.231906673	0.098797507	0.144116666	0.006302263
PAES	0.604658346	0.554848878	0.343459556	0.156123736	0.160591451	0.084886574

Table 5.5: **Hyper-volume**

Hyper-Volume(SD)						
	I10-5g5	I10-5p	I20-15g10	I30-5g5	I30-15g5	I40-20g10
MMOBA	0.058786138	0.031745826	0.032141193	0.144163518	0.07415734	0.090940579
DBEA	0.026343469	0.021529587	0.117248495	0.129655806	0.190550324	Nan
MOEAD	0.142657155	0.170431685	0.201025656	3.44062E-15	0.20128042	0.140783647
NSGAII	0.083238988	0.028936093	0.095137706	0.155451169	0.059633769	0.18785326
OMOPSO	0.18742501	0.06633796	0.139930066	0.028876681	0.110940426	0.268546412
PESA2	0.101759263	0.125746567	0.091735591	0.253698366	0.135250059	8.18024E-15
Random	8.72278E-15	3.25247E-15	0.131979778	0.08807860	0.109289946	0
RVEA	0.208758193	0.19431956	0.259284035	0.133439756	0.268074848	0.107056744
SPEA2	0.06018822	0.035578262	0.101456236	0.07208568	0.141537403	6.67058E-16
VEGA	0.104173169	0.02634606	0.001530498	0.239665359	0.070645633	0.066036915
e-MOEA	0.039070589	0.042832523	0.196479095	0.098369618	0.185839253	NAN
GDE3	0.281860182	0.129785792	0.204248806	NAN	1.40433E-16	2.96616E-15
SMPSO	0.005607256	0.120383314	0.315755167	0.282843323	0.189358578	0.00869051
PAES	0.059097369	0.078300588	0.097406697	0.134351381	0.234646171	0.126080192

Table 5.6: **Hyper-volume**

On calculating hyper-volume of each algorithm on all 6 datasets, the tables $5.5, 5.6$ clearly shows that our proposed modified multi-objective bat algorithm (MMOBA) performed better in case of large datasets i.e. i20-15g5, i30-5g5, i30-15g5 and i40-20g10, while DBEA which is which is simple decomposition based evolutionary algorithm on small datasets i.e. i105g5 and i105p7. However results of proposed algorithm on small datasets are almost near to the results of DBEA on small datasets.

Spread						
	I10-5g5	I10-5p7	I20-15g10	I30-5g5	I30-15g5	I40-20g10
MMOBA	0.70420331	0.703925755	**0.656440881**	**0.743523064**	**0.615985304**	**0.627866516**
DBEA	0.68041387	0.659575133	0.676362468	0.819449519	0.659975466	NAN
MOEAD	**0.220653054**	**0.422467783**	1.097801132	1.00636496	0.993005838	1.009314301
NSGAII	1.124432401	0.903098958	1.010462781	0.753519053	0.85166715	1
OMOPSO	1.006118333	0.97256788	0.93150407	0.989795967	1	1
PESA2	0.989873818	1.073357029	1.017264468	0.953332753	1	1
Random	1	1	1	1	1	1
RVEA	0.734196749	0.917081137	1	1	1	1
SPEA2	0.919840128	0.938549277	0.980877033	1.014359983	0.889103007	1
VEGA	1	1	1	1	1	1
e-MOEA	0.806048053	0.871285558	0.710611363	0.98473582	0.721073042	NAN
GDE3	0.933850963	0.918154013	0.926159182	NAN	1	1
SMPSO	0.906137383	0.980399465	0.96906604	1.008349223	1	1
PAES	0.76638296	0.797171309	1.050428005	0.974579897	0.931393281	1

Table 5.7: **Spread**

Spread is another parameter for optimization problems that specifies the rate in which solutions are dispersed in front. According to the results presented in table 5.7 shows that our proposed modified multi-objective bat algorithm (MMOBA) performed better here also in case of large datasets i.e. i20-15g5, i30-5g5, i30-15g5 and i40-20g10, while MOEAD which is also a decomposition based evolutionary algorithm on small datasets i.e. i105g5 and i105p7.

	GD					
	I10-5g5	I10-5p7	I20-15g10	I30-5g5	I30-15g5	I40-20g10
MMOBA	0.044302538	0.025020041	0.050803325	0.127688001	**0.073605108**	**0.096399854**
DBEA	**0.003352854**	0.005242017	0.097564746	0.11783803	0.138309034	NAN
MOEAD	0.036192937	0.073544207	0.048071368	6.506527962	1.204537192	0.182451911
NSGAII	0.067220275	0.159536341	0.143388325	0.166374633	0.136923916	0.373708666
OMOPSO	0.019160655	0.015826623	1.028429655	0.33616486	0.902741519	0.461976173
PESA2	0.072032886	0.171946474	0.34018374	0.305206434	1.588701646	0.795187153
Random	1.592888926	0.897403288	0.785394077	0.228350562	0.930765382	NAN
RVEA	0.044716774	0.532163252	0.442633728	0.515690665	0.517195908	0.64118836
SPEA2	0.043837898	0.052467676	0.04837557	**0.069041055**	0.088460055	1.329569132
VEGA	0.212510421	1.375278151	1.565226435	0.279682567	0.47539022	0.350282009
e-MOEA	0.003380314	**0.002743898**	0.088823607	0.077247485	0.17574166	NAN
GDE3	0.268206159	0.240574774	0.051115774	NAN	1.255004619	1.041190974
SMPSO	7.738994175	0.04711882	0.201425044	3.019125959	0.627074074	0.137729377
PAES	0.005446067	0.007113635	**0.04671419**	0.082226864	0.172949844	0.693142082

Table 5.8: **Generational distance**

	IGD					
	I10-5g5	I10-5p7	I20-15g10	I30-5g5	I30-15g5	I40-20g10
MMOBA	0.064946235	0.029630163	0.057224173	0.098783269	**0.079187524**	**0.094937838**
DBEA	0.006267917	0.004851514	0.104153759	0.101320495	0.118851448	NAN
MOEAD	0.065451948	0.110796879	0.10592118	2.165306191	0.583772229	0.140873236
NSGAII	0.125895019	0.061182844	**0.05667836**	0.164741527	0.124658262	0.603596822
OMOPSO	0.02365667	0.013082904	0.5367735	0.248422294	0.832263498	0.643310781
PESA2	0.054316026	0.049499329	0.263425513	0.188643028	1.226000817	0.912602637
Random	1.478102022	1.138845791	0.677011529	0.33527847	0.839465171	NAN
RVEA	0.058890206	0.511423534	0.587612229	0.574835282	0.664715939	0.6398338
SPEA2	0.063346955	0.03196065	0.044882713	0.097353088	0.121609297	1.397116422
VEGA	0.063346955	0.03196065	0.044882713	0.097353088	0.121609297	1.397116422
e-MOEA	**0.004779952**	**0.004106833**	0.09475331	0.099680515	0.145601432	NAN
GDE3	0.133424938	0.084877169	0.067589153	NAN	1.328444904	1.118818202
SMPSO	17.86128095	0.037878979	0.234278684	1.890655616	0.576516465	0.286859179
PAES	0.006841269	0.012575354	0.071146869	**0.080177671**	0.19806489	0.531814873

Table 5.9: **Inverted Generational distance**

On calculating generational and inverted generational distance of software project scheduling problem on set of algorithms, results in table 5.8 and 5.9 shows that proposed algorithm performed better on extreme large datasets i.e. i30-15g5 and i40-20g10, while DBEA, e-MOEA, PAES, NSGAII and SPEA2 gives better generational and inverted generational distance on other datasets.

5.3 Pareto Front

Figure below represents reference Pareto fronts of all algorithms gathered on all 6 datasets.

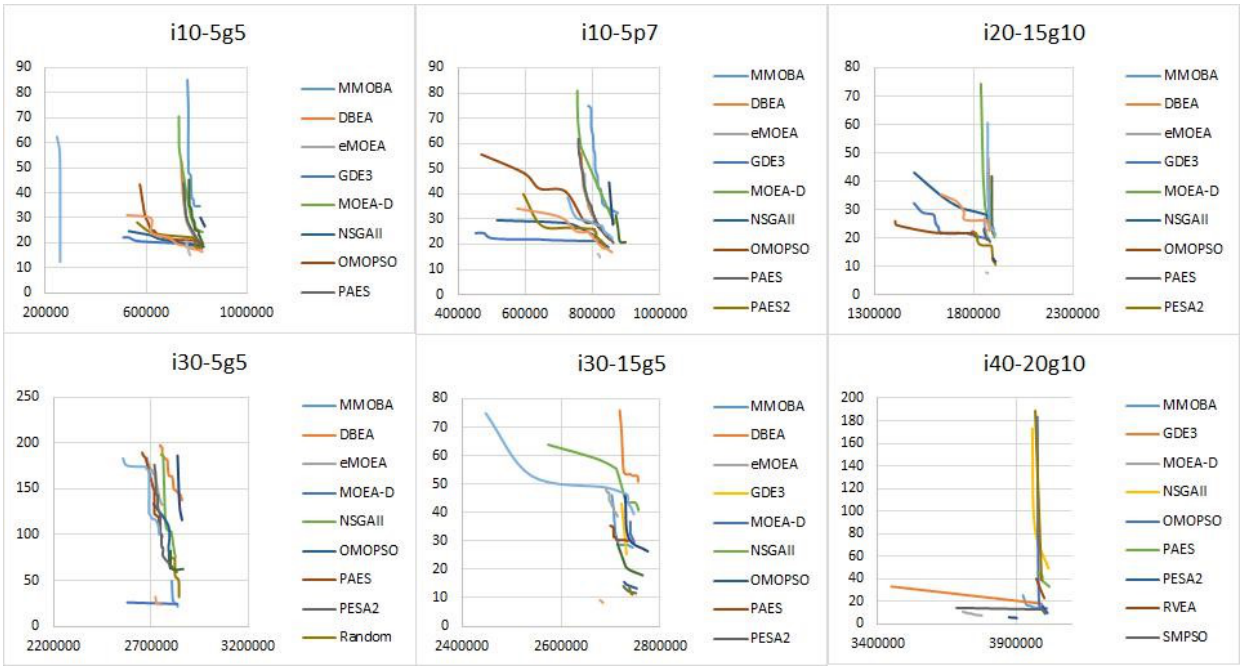

Figure 5.1: **Reference Pareto fronts of all datasets**

6 Conculsion

In this thesis algorithms from MOEA frameworks are used for comparison with our proposed algorithm modified multi-objective bat algorithm (MMOBA) with 6 different sized datasets and it is concluded that proposed modified multi-objective bat algorithm (MMOBA) performed better than most of the other evolutionary algorithms on large datasets of software project scheduling problem. Hyper-volume and spread results showed that proposed algorithm worked better for large data sets, generational and inverted generational distance showed that for extreme large datasets proposed algorithm worked better while DBEA and MOEA-D performed better for small datasets. For future research proposed bat algorithm can be merged with DBEA and MOEA-D and hybrid algorithm can be made for getting better results for all large and small datasets.

Bibliography

[1] Ai-Junkie. ["Genetic Algorithms in Plain English"] .
http://www.ai-junkie.com/ga/intro/gat1.html

[2] Chun [*"Genetic Algorithms"*].
https://www.doc.ic.ac.uk/ nd/surprise$_9$6/$journal/vol1/hmw/article1.html$

[3] Mitchell, Melanie (1996). ["An Introduction to Genetic Algorithms"]. Cambridge, MA: MIT Press. ISBN 9780585030944.

[4] Tutorialspoint. [*"Genetic Algorithms Fundamentals"*]
https://www.tutorialspoint.com/geneticalgorithms/

[5] Manish Dixit, Nikita Upadhyay and Sanjay Silakari.[*"An Exhaustive Survey on Nature Inspired Optimization Algorithms International Journal of Software Engineering and Its Applications"*]. Vol. 9, No. 4 (2015), pp. 91-104

[6] Dorigo M, Maniezzo V, Colorni A, [*" feedback as a search strategy"*]. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

[7] Dorigo M, Maniezzo V, Colorni A. [*"Ant System: Optimization by a colony of cooperating agents"*]. IEEE Trans Syst Man Cybernet Part B 1996; 26 :29-41

[8] Christian Blum. [*"Ant colony optimization: Introduction and recent trends"*]. Physics of Life Reviews 2 (2005) 353-373.

[9] AnirudhShekhawat, Pratik Poddar, Dinesh Boswal. [*"Ant colony Optimization Algorithms : Introduction and Beyond"*]. Artificial Intelligence Seminar 2009.

[10] Wikipedia. [*"Ant colony optimization algorithms"*]. 26 May 2017.
https://en.wikipedia.org/wiki/Ant-colony-optimization-algorithms

[11] Anonymous.[*"MODELING OF MULTI-OBJECTIVE TASK SCHEDULING PROBLEM IN CLOUD ENVIRONMENT"*]. http://shodhganga.inflibnet.ac.in/bitstream/10603/8273/23/12_chapter%202.pdf

[12] Wikipedia.["Complexity class"].24 May 2017.
https://en.wikipedia.org/wiki/Complexity-class

[13] Weisstein, Eric W. MathWorld–A Wolfram Web Resource. ["NP-Problem"].
http://mathworld.wolfram.com/NP-Problem.html

[14] Xin-She Yang. ["Firefly Algorithms for Multimodal Optimization"]. Springer-Verlag Berlin Heidelberg 2009 pp. 169-178, 2009.

[15] IztokFister, Xin-She Yang, IztokFister Jr. and Janez Brest. ["A comprehensive review of firefly algorithms"]. Swarm and Evolutionary Computation.

[16] SupriyaShilwant.Slideshare. ["Firefly Algorithm"].
https://www.slideshare.net/supriyashilwant/firefly-algorithm-49723859

[17] Xin-She Yang and Suash Deb. ["Cuckoo Search via Levy Flights"]. IEEE 2009 World Congress on Nature Biologically Inspired Computing.

[18] Payne R. B, Sorenson M. D., and Klitz K., ["The Cuckoos"]. Oxford University Press,(2005)

[19] Wikipedia.["Cuckoo Search"]. 25 May 2017.
https://en.wikipedia.org/wiki/Cuckoo-search

[20] James Kennedy and Russell Eberhart. ["Particle Swarm Optimization"]. 1995 IEEE pg1942-1948

[21] S. Rajasekar, P. Philominathan and V. Chinnathambi. ["RESEARCH METHODOLOGY"]. Corenell University arXiv.org, physics.

[22] Anonymous. ["An Instance Generator for the Project Scheduling Problem"]. April 1st of 2005.
http://tracer.lcc.uma.es/problems/psp/generator.html

[23] Xin-She Yang. ["A New Metaheuristic Bat-Inspired Algorithm"]. Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK.

[24] Xin-She Yang.["Bat Algorithm: Literature Review and Applications"]. Int. J. Bio-Inspired Computation, Vol. 5, No. 3, pp. 141-149 (2013).

[25] T. Hanne and S. Nickel.["A Multi-Objective Evolutionary Algorithm for Scheduling and Inspection Planning in Software Development Projects"]. Berichte des Fraunhofer ITWM,Nr. 42 (2003).

[26] Enrique Alba and J. Francisco Chicano. ["Software project management with GAs"].Information Sciences 177 (2007) 2380-2401

[27] D.Sundar, B.Umadevi and Dr.K.Alagarsamy. ["Multi Objective Genetic Algorithm for the Optimized Resource Usage and the Prioritization of the Constraints in the

Software Project Planning"]. International Journal of Computer Applications (0975 - 8887) Volume 3 - No.3, June 2010.

[28] Francisco Chicano, Francisco Luna, Antonio J. Nebro and Enrique Alba.["Using Multi-Objective Metaheuristics to Solve the Software Project Scheduling Problem"]. GECCO'11,July 12-16, 2011, Dublin, Ireland.

[29] ConstantinosStylianou and Andreas S. Andreou. ["Intelligent Software Project Scheduling and Team Staffing with Genetic Algorithms"].Part II, IFIP AICT 364, pp. 169-178, 2011.

[30] Milena KarovaNevenaAvramova. ["A Genetic Algorithm Basic Approach for Software Management Project"]. International Conference on Computer Systems and Technologies - CompSysTech'12.

[31] Leandro L. Minku, Dirk Sudholt and Xin Yao. ["Evolutionary Algorithms for the Project Scheduling Problem: Runtime Analysis and Improved Design"]. GECCO'12, July 7-11, 2012, Philadelphia, Pennsylvania, USA.

[32] Abel Garcia-Najera and Mariadel Carmen Gomez-Fuentes. ["A Multi-Objective Genetic Algorithm for the Software Project Scheduling Problem"]. Av. Vasco de Quiroga 4871,Col. Santa Fe Cuajimalpa, Mexico, D.F., 05300, Mexico 2014

[33] Leandro L. Minku, Dirk Sudholt and Xin Yao. ["Evolutionary Algorithms for the Project Scheduling Problem: Runtime Analysis and Improved Design"]. GECCO'12, July 7-11, 2012, Philadelphia, Pennsylvania, USA.

[34] MayowaAyodele, MayowaAyodele and Olivier Regnier-Coudert, ["Probabilistic Model Enhanced Genetic Algorithm for Multi-Mode Resource Constrained Project Scheduling Problem"]. GECCO '15 July 11-15, 2015, Madrid, Spain.

[35] ConstantinosStylianou and Andreas S. Andreou. ["Investigating the impact of developer productivity, task interdependence type and communication overhead in a multi-objective optimization approach for software project planning"]. Advances in Engineering Software 98 (2016) 79-96.

[36] Jing Xiao, Xian-Ting Ao and Yong Tang. ["Solving software project scheduling problems with ant colony optimization"]. Computers Operations Research 40 (2013) 33-46

[37] Wei-Neng Chen and Jun Zhang ["Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler"]. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 39, NO. 1, JANUARY 2013.

[38] Broderick Crawford , Ricardo Soto, Franklin Johnson and Eric Monfroy. ["Ants Can Schedule Software Projects"]. C. Stephanidis (Ed.): Posters, Part I, HCII 2013, CCIS 373, pp. 635-639, 2013

[39] Bharti Suri, PoojaJajoria. ["Using Ant Colony Optimization in Software Development Project Scheduling"]. 2013 International Conference on Advances in Computing,Communications and Informatics (ICACCI).

[40] OlfaDridi, SaoussenKrichen and Adel Guitouni. ["A multiobjective hybrid ant colony optimization approach applied to the assignment and scheduling problem"]. Intl. Trans. in Op. Res. 00 (2014) 1-19.

[41] Broderick Crawford, Ricardo Soto, Franklin Johnson , Fernando Paredes and Miguel Olivares Suarez. ["Max-Min Ant System to solve the Software Project Scheduling Problem"]. Expert Systems with Applications.

[42] Broderick Crawford, Ricardo Soto, Franklin Johnson , Fernando, Sanjay Misra and Eduardo Olguin. ["SOFTWARE PROJECT SCHEDULING USING THE HYPERCUBE ANT COLONY OPTIMIZATION ALGORITHM"]. Tehnickivjesnik 22, 5(2015), 1171-1178.

[43] Jing Xiao, Mei-Ling Gao and Min-Mei Huang. ["Empirical Study of Multi-Objective Ant Colony Optimization to Software Project Scheduling Problems"]. GECCO '15, July 11 - 15, 2015, Madrid, Spain.

[44] A. C. Biju, T. Aruldoss Albert Victoire, and KumaresanMohanasundaram. ["An Improved Differential Evolution Solution for Software Project Scheduling Problem"]. The Scientific World Journal Volume 2015, Article ID 232193, 9 pages.

[45] Broderick Crawford, Ricardo Soto, Franklin Johnson , Fernando and Sanjay. ["The Use of Metaheuristics to Software Project Scheduling Problem"]. ICCSA 2014, Part V, LNCS 8583, pp. 215-226, 2014.

[46] Xiaoxia Huang, Tianyi Zhao and ShamsiyaKudratova. ["Uncertain mean-variance and semi-variance models for optimal project selection and scheduling"]. Knowledge-Based Systems 93 (2016) 1-11

[47] XiuliWu , Pietro Consoli , Leandro Minku , Gabriela Ochoa and Xin Yao. ["An Evolutionary Hyper-heuristic for the Software Project Scheduling Problem"]. Springer-Verlag Berlin Heidelberg 2016.

[48] Broderick Crawford, Ricardo Soto, Franklin Johnson, Carlos Valencia and Fernando Paredes. ["Firefly Algorithm to Solve a Project Scheduling Problem"]. Springer International Publishing Switzerland 2016.

[49] Anil K. Gupta, Ken G. Smith and Christina E. Shalley.["The Interplay Between Exploration and Exploitation"]. Academy of Management Journal 2006, Vol. 49, No. 4, 693-706.

[50] Christian ARTIGUES.["The Resource-Constrained Project Scheduling Problem"]. http://www.iste.co.uk/data/doc-dtalmanhopmh.pdf

[51] Kennedy, J. and R. Eberhart, 1995. ["Particle swarm optimization"]. Proceedings of the IEEE International Conference on Neural Networks, Piscataway, pp: 1942-1948

[52] RoohollahKalatehjari. ["The Contribution of Particle Swarm Optimization in Three-Dimensional Slope Stability Analysis"]. The Scientific World Journal In Press(10) June 2014.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 4. March 2019