A Comparative Study of the Relative Performance and Real-World Suitability of Optimization Approaches for Human Resource Allocation

Sultan M A Al Khatib

A Thesis Submitted for the Degree of

Doctor of Philosophy

At the University of East Anglia

September 2018

A Comparative Study of the Relative Performance and Real-World Suitability of Optimization Approaches for Human Resource Allocation

Sultan M A Al Khatib

Abstract

The problem of Staffing and Scheduling a Software Project (SSSP), where we consider Human Resource Allocation (HRA) to minimize project time, offers a management challenge for Project Managers (PM's). Unlike the general HRA problem, SSSP involves determination of the assignment of a fixed amount of resources to teams and the allocation of these teams to project's jobs. SSSP problem arises across a diverse range of resources' and project characteristics (discrete variables), and this variety has offered a wide range of HRA methods.

The general consensus is that the benchmark for SSSP are Meta-heuristic optimization techniques using deterministic or stochastic simulation of time. However, different HRA methods and project attributes are considered by SSSP approaches, and their solutions need to be compared against each other. The majority of SSSP approaches provide their approximation using Genetic Algorithm (GA) validated by a synthetic data or empirical method such as Quasi-experiment. Limited studies offer the comparison between these SSSP approaches, either by a comprehensive survey or systematic literature review for qualitative concepts.

We aim to answer a set of research questions including: what is the best way to show the quality and performance differences between SSSP approaches? And, are these SSSP approaches suitable for industrial adoption? Our thesis is that the best methodology is to identify according to the conceptual models used by the approaches a set of challenging data levels. In support of our thesis, we propose a systematic benchmarking and evaluation approach that encompass the data levels, and a set of quality measures. Next, we propose an empirical study that assess how PMs from software industry perform the allocation given the same datasets. The results of both works demonstrate significant differences between the approaches, highlighted four methods that advances the research filed, and provide interesting discussion on the PMs' practices on SSSP.

List of Abbreviations

Abbreviation	Meaning
COCOMO	Constructive Cost Model
CP	Critical Path
CPM	Critical Path Method
CRD	Critical Resource Diagram
CT	Computation Time
DAG	Directed Acyclic Graph
DM	Decision Maker
DP	Dynamic Programming
DTBP	Dynamic Team with Binary Participation method
DTPR	Dynamic Team with Participation Rate method
EPT	Estimated Project Time
FP	Function Point
GA	Genetic Algorithm
GUI	Graphical User Interface
HC	Hill Climbing
HRA	Human Resource Allocation
KLOC	Kilo Line Of Code
MAAPE	Mean Arctangent Absolute Percentage Error
MoGA	Multi-objective Genetic Algorithm
NSGAII	Non-dominated Sorting Genetic Algorithm II
PERT	Program Evaluation and Review Technique
PM	Project Manager
PMBOK	Project Manager Body Of Knowledge
PSO	Particle Swarm Optimization
PSP	Project Scheduling Problem
RCPS	Resource Constrained Project Scheduling Problem
SA	Simulated Annealing
SBSE	Search Based Software Engineering
SBSPM	Search Based Software Project Management

SE	Software Engineering
SLOC	Source Line Of Code
SPM	Software Project Management
SSSP	Staffing and Scheduling a Software Project
STQS	Static Team with Queueing Simulator method
STTS	Static Team with Time Simulator method
SWECOM	IEEE Software Engineering Competency Model
TPG	Task Precedence Graph
WBS	Work Breakdown Structure
WP	Work Package

Table of Contents:

List of Tables

List of Figures

Chapter 1 Introduction

In this chapter, an overview of human resource allocation in software projects, including a general background on related topics of optimization approaches, is presented in Section 1.1. In addition, a general formalization of human resource allocation with consideration to project time minimization problem is presented in Section 1.2. This chapter also presents our motivation, aims, and research questions in Sections 1.3 and 1.4, respectively. Section 1.5 provides to the reader an overview about the research methodology and process carried out for the work for this thesis. Section 1.6 lists the contributions of this thesis, and the thesis organization is presented in Section 1.7.

1.1. Human Resource Allocation in Software Projects

Human Resource Allocation (HRA) can be defined as the process of determining a feasible and optimal schedule for a set of jobs according to the resources' availability and/or the completion time of these jobs target(s) [1]. It is the responsibility of a Project Manager (PM) to perform HRA given the interdependent relationship between human resources and jobs, which requires the PM to identify which job should be done by whom with careful selection of the competent resources [2].

HRA is a vital and crucial part of project management which plays a critical role in maintaining project outcomes to the planned constraints of quality, cost, and time. Finding the optimal resource allocation plan with respect to project schedule and maintaining it with high quality, low cost, and minimum time standards is a complex problem for a PM to solve in reasonable time given the limitation of deploying and delivering the final product. It has been proven that inadequate human resource planning is one of the causes of failure especially in software projects [2].

Project management as defined by Project Management Body Of Knowledge (PMBOK) is the "application of knowledge, skills, tools and techniques to project activities to meet the project

requirements" (p. 6) [3], and it is not a surprise that more skilled, knowledgeable, and expert PMs are always in demand by many organizations. Moreover, project management is the most dynamic and vibrant among the management disciplines [4], and in software production several management activities are critical for success, however these activities potentially have conflicting goals with each other [5]. This is due to the nature of software construction and the characteristics of software projects, which predominantly involve human resources and requires cognitive processes of individuals collaborating in teamwork to create the software [4]. For example, it is hard to balance between project time span and product quality when different skills are heavily involved in the software construction, testing and quality assurance, while at the same time availability of these skills amongst the resources is scarce.

There are several characteristics that make software production and it related management activities differ from any other projects. Software is an intangible product that is expected to provide a unique solution. However, with its intangible nature, stakeholders often provide imprecise and/or incomplete information about the required solution. Product scope could then have some potential changes if these imprecise or incomplete requirements are not rectified. Moreover, software production involves complex development of interfaces and core systems, and at the same time it often has dependability and interaction with other software(s), hardware(s), and processes, which requires continuous updating of practices according to the constant evolution of processes, methods, and tools. This nature of software requires different types of testing to ensure software quality and security however with this complex nature and imprecise requirements, quality measures can be a hard target for software engineers to achieve. In addition, software production is a human-based development. Therefore, individuals with intensive intellectual capital are heavily required to form teams, where the members of these teams have to have their communication and coordination as clear as possible [4]. Thus, software projects differ by its nature from any other projects, and it is not a surprise that in software projects the management activities and goals are the major concern rather the technical ones [6].

One of the advantages of improving human resource allocation is to minimize project time by which software firms can be more productive [7]. Many studies have presented the importance of minimizing software project time among the management goals of time, cost, and quality as in [5] [8], and as stated by [9] with industrial evidence that for standard software development the organizations have in their highest priority to minimize project schedule depending on the availability of skills and expertise as a way of reducing time to market.

In early work on project time, different models are proposed for time estimation and management such as Work Breakdown Structure (WBS), Program Evaluation and Review Technique (PERT), Critical Path Method (CPM), etc. -see Section 2.1.2- however none of these models support the resources attributes and the dynamic nature of resource allocation taking into account team aspects. To solve different Software Engineering (SE) problems including software project time minimization, optimization techniques (Search-Based Algorithms) are employed by many researchers, and the dawn of a new field was born and coined by the term Search-Based Software Engineering (SBSE) by [10]. SBSE focuses on the application of special optimization techniques that belong to class Meta-Heuristic to different software engineering problems -see Section 2.2 -. Part of the earliest work in SBSE described by [11] are both in [12] and [13], where [12] have employed Genetic algorithm -see Section 2.2- to determine the best resource allocation for software project management. The one in [13] on the other hand was the first to formalize requirement planning into a Next Release Problem (NRP) and provide three approaches using Greedy, Hill Climbing, and Simulated Annealing optimization techniques -see Section 2.2- to maximize satisfaction of a selected stakeholder(s) on requirement prioritization for the next release.

A more advanced classification is introduced by [5] as a subfield of SBSE named Search-Based Software Project Management (SBSPM), and comprehensively surveyed for the approaches that employ optimization techniques on Software Project Management (SPM) problems including Software project time minimization. Limited number of approaches are found by [5] that have explored the nature of software project time minimization and provided approximation methods for it using an optimization technique.

HRA with consideration to project time span minimization is an optimization problem for a cost function that involves constraints on project tasks and the available resources. For instance, precedence relationship between partial or entire project tasks, and/or skills between tasks and resources are required to be satisfied. These constraints however can be soft or hard by which violation of a soft constraint can be acceptable and a process is put to rectify that by applying penalties, or for a hard constraint any solution that violate it should be omitted. HRA combines two aspects which are: a) the workflow of jobs that the resources have to follow implemented by a graphical representation such as Directed Acyclic Graph (DAG), and b) criteria for resource selection (e.g. skills) [2].

Different representations of HRA problem have been introduced by different incarnations depending on the resources' type addressed in the problem such as human, machines, etc. An early version of HRA problem has been introduced as a Resource-Constrained Project Scheduling problem (RCPS), which tend to tackle several kinds of resources [14]. A related problem to this has been later tackled by many approaches as in [14, 15] named Project Scheduling Problem (PSP), which only considers human resources and their skill(s). Given the nature of HRA in software projects which is not only concerned about project schedule, but in addition the resources staffing and their productivity and distribution to teams as in [16], our general optimization problem addressed for this thesis is represented by the following Section 1.2.

1.2. Staffing and Scheduling a Software Project

Staffing and Scheduling a Software Project (SSSP) is one of the software project management problems. This problems is associated with exploring a set of possible solutions and searching for the best minimized project time span among the feasible solutions. However, the search space of possible candidate solutions for this problem is typically large and requires extensive processing time to find the best one [17]. This problem is well-known to be NP-Hard class of computational complexity that no known algorithm can find an optimal solution for it in a polynomial time [18]. Here the optimization techniques are used to help in this particular management task aiming to produce optimal or near optimal solutions within a reasonable computational time. SSSP problem can be represented by five main elements. These elements are depicted in the following Figure 1.

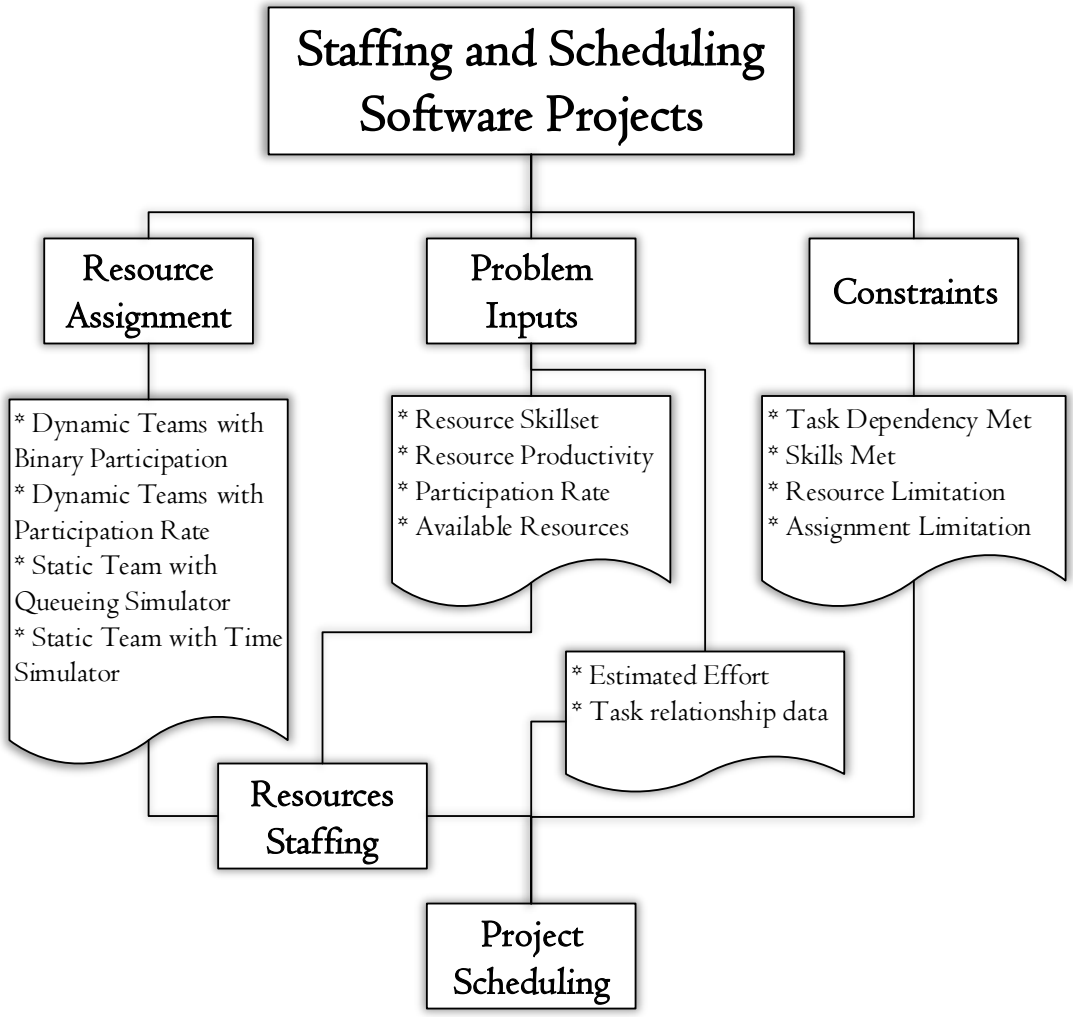

Figure 1: SSSP Elements

From Figure 1, it can be seen that the SSSP problem requires the identification of three main elements. These elements are problem inputs, resource assignment method, and the software project constraints. In addition, SSSP problem includes two nested problems, which are resource staffing and project scheduling. As can be seen in Figure 1, resource staffing problem is based on the inputs of resources' availability, properties, and the way of assigning these resources to teams. Once staffing of resources is completed, project scheduling can be then established based on the outcome of resource staffing and additional two aspects. These aspects are the second part of the problem inputs of project properties and the project constraints of task dependency, skills, etc. which have to be adhered to within the outcome schedule. According to these elements the optimization problem of SSSP can be mathematically formulated as follow.

Problem Formalization:

Staffing the resources and scheduling the tasks of a software project can be formulated as an optimization problem of a cost function for minimizing project time span pT as follow.

$$min\, f(pT)$$

$$where,$$

$$pT = max\{CP_1, CP_2, \dots, CP_\gamma\}$$

$$CP_d = \sum_{k=1}^{I} S(t_{d_k})$$

$$S(t_l) = e_{t_l} / \sum_{r=1}^{n} pro_r * Q(t_l, r)$$

$$subject\ to,$$

$$\forall\, r, t_l : Q(t_l, r) = 1\ \exists\ C_r \cap C_{t_l} \neq \phi$$

$$\sum_{r=1}^{n} Q(r, t_l) \leq b\ , where\ b \in \mathbb{Z}^+$$

$$\forall\, t_l \in T\ \exists\ dp_l \in TD : dp_l\ \subseteq FT$$

Software project time span pT can be defined as the maximum Critical Path CP length among the set of alternative CPs defined for project schedule. With γ number of CPs, the length of a CP_d involving I number of tasks can be determined by the summation of estimated time S of each task t_{d_k}. Time S of task t_l can be calculated by the division of estimated effort e of t_l over the overall productivity of the resources assigned to it. These resources who are assigned to task t_l can be identified according to decision variable Q which will return a value of 1 for $Q(t, r)$ if resource r is assigned to task t, or 0 otherwise. The return value of variable Q will be then used to identify which resource productivity pro will be add to the overall value. However, the identification process of the software project schedule time should comply with a set of constraints. Each resource r possesses a set of competencies C and for each r assigned to t_l, r should possesses the required competencies C for task t_l represented as:

$$\forall\, r, t_l : Q(t_l, r) = 1\ \exists\ C_r \cap C_{t_l} \neq \phi$$

In addition, the number of resources participating to perform one task should not exceed the limit b value. For this constraint penalty should be applied as the overhead communication is anticipated to reduce the team's productivity and the development speed.

$$\sum_{r=1}^{n} Q(r, t_l) \leq b \,, where\ b \in \mathbb{Z}^+$$

The precedence relationship should be satisfied so that for each task t_l defined in the project tasks set T its predecessors must be finished in order t_l to be started. The dependency information can be obtained by the set TD, which hold a subset task dp for each task t_l. Subset dp_l accordingly hold the information about task t_l predecessor(s).

$$\forall\ t_l \in T\ \exists\ dp_l \in TD :\ dp_l\ \subseteq FT$$

1.3. Motivation

Since the early work presented in [12], the field of software project HRA optimization has gradually become more advanced, and many models have been proposed in this field over the last three decades, where each has potentially demonstrated a real-world allocation problem according to the targeted organization environment. However, evidences from real-world examples provide diversity of human resource allocation problems described by [9] and [19].

Different approaches have been proposed for SSSP problem. These approaches employ Meta-Heuristics and each is targeting specific project and resource properties based on different perspectives. One of these perspectives assumes that the resources share similarity in skills and productivity as in [20, 21]. Based on this assumption they have formed their HRA problem into a queueing system to distribute the tasks to different teams, where the formation of teams only depends upon the number of resources, so the more resources you have, the more you likely to finish the work earlier.

Another proposal has shaped SSSP by considering the distribution of resources into different project tasks with the assumption that resources can only be allocated with a percentage of their daily working time. This percentage type of allocation requires the identification of participation rate for each resource to each task. These approaches have made their assumption where resources are differing in terms of skills but they share same productivity as in [14, 18, 22, 23].

According to the description presented by the optimized SSSP approaches, we have identified four methods of resource and team assignment based on the concepts of dynamic and static team formation, time and queueing assignment simulation, as well as participation of resources. These assignment methods can be represented as a categorization for the optimized SSSP approaches represented by the following Table 1.

Table 1: HRA and Team Assignment Methods

	Method
1	*Static Teams with Queue Simulator (STQS)*
2	*Static Teams with Time Simulator (STTS)*
3	*Dynamic Teams with Binary Participation (DTBP)*
4	*Dynamic Teams with Participation Rate (DTPR)*

For full details about the team assignment methods the reader can refer to Chapter 5, and Chapter 2 for specification about the selected SSSP approaches that comply with these methods.

On the other hand, skills are not always the only best choice to use for optimal resource allocation in software projects [16]. More factors and aspects are involved in determining the fitness of resources to project tasks such as resources' productivity. While many approaches have assumed that productivity of software project resources is always similar to each other, others have demonstrated how this factor could be a key role in reducing the search time while relaxing skills constraint. These approaches that consider the differences between resources in terms of skills, and productivity are limited as in [23]. Putting all these assumptions into practice requires demonstration of which can lead to a better solution. It is understandable that all these assumption can be seen in the industry practice, however, it is important for us to understand which and why each of these approaches has the potential of industrial adoption. Throughout the literature -see Section 2.3-, we have found that limited surveys and systematic literature reviews have been performed on the approaches that tackle SSSP problem as in [5] and [24], and none provides evaluation and comparison of the runtime results between these approaches except the one in [15].

We believe that providing a comprehensive evaluation and comparison between SSSP approaches can help on moving this field of research one step towards the industrial adoption. In addition, one potential work that can be added to this is an empirical evaluation of how PMs from software industry can perform HRA on the same data used for the approaches evaluation. Moreover, it is also important to capture which software project and resource's aspects are important for PM's to consider. While PMs are in urgent need for good quality and accurate software project planning and estimation techniques, the discussion in [11] argues that this will keep the SBSE community attention and interest in this subject for more work on management plan robustness, and integration of software engineering and management activities.

1.4. Research Aims and Questions

Our main aim in this thesis is to provide a complete study that accumulates the findings from software project time span minimization including SSSP approaches performance while showing how to address their problem formulation, measure their outcomes quality, and express the findings from these approaches compared against each other. It is important to see how the SSSP problem has been addressed by different incarnations, and whether these incarnations are sharing similarity between each other in terms of the allocation method, and the software project and resource's attributes used in their problem formulation. One of the targets of this aim is to standardise the experiments under one objective of the SSSP problem defined in Section 1.2, of project time span minimization. The main reasons to adopt the time minimization for our work is twofold. While part of SSSP approaches involves multi-objective optimization, the comparison between these approaches requires unity of common optimization objective(s), in which the time is the only common one amongst them all. In addition, evidence by [24] shows how the mainstream of SSSP approaches are considering project completion time for minimization with 44 approaches out of 52.

Our main focus is to compare between the different optimized solutions (approaches) for software project HRA, where each consider and adopt an optimization technique, problem variables, setting, and adjustable stochastic process to approach the SSSP problem. It is important to note that this research is not about comparison between the optimization techniques. For studies that consider the comparison for exact optimization techniques we refer to [25], and for different stochastic, and heuristic techniques we refer to [21, 22, 26]. In addition, the reader can refer to [27], which provides a comparison between different project management tools. From [27], it can be seen that the allocation of resources is the least to be considered by the management tools as they only provide partial and non-automatic assignment of resources regardless of the management objectives.

The nature of this research as in other fields of study has some limitations. Lack of industrial contribution of historical data to be extracted, or time availability of project managers to share their opinion and expertise on particular subject(s) are the main limitations on software project HRA research. In addition, there are some obstacles, due to the competitive market and/or sensitivity of data, on conducting meetings with representatives from the industry while agreements have to be made by both sides for confidentiality requirement and non-disclosure, which are vital for a research to progress. Despite all these limitations our second aim is to

empirically capture and evaluate the current industry practices and the main software project attributes that are important for PM to consider.

From both aims, a set of main questions can be formulated by which we need thoroughly from their answers to acquire understanding, explore subject(s), research the field and trends, and report findings related to SSSP problem, and software project time span minimization. These questions are:

- Is there an automated SSSP approach that reliably solves the SSSP problem?
- Do these approaches outperform expert intuition in solving the SSSP problem?
- Do these approaches reflect the software projects and project managers' real needs?

These three questions form the roadmap for the work carried out for this thesis. Throughout the following chapters, some follow-up questions will also be identified as the exploration process of SSSP problem, approaches, and current industry practices are gradually moving forward. For example, a follow-up question will be formulated in a later chapter to ask how better to investigate the automated SSSP approaches' performance and quality. These questions will help us to understand if there are any differences between the SSSP approaches in terms of the runtime outcomes. These follow-up questions can be highlighted by the following:

1. What are the differences between SSSP approaches?
2. Why do these differences occur?
3. Are these approaches and their proposed data to use suitable for the software industry?
4. How would an industrial setting representative use particular project data to provide a solution for SSSP problem?

1.5. Overview of our Methodology and Benchmarking Approach

Broadly speaking, benchmarking in SBSE can be performed for a single approach by employing different optimization techniques and a base for the benchmark as suggested by [10, 17] using a random search for comparison. SSSP approaches have evaluated their proposals either by comparing their solution with different optimization techniques as in [21, 22] or empirically with a single industrial partner as in [23, 28], using a real-world datasets as in [20-22] or synthetic one as in [14, 15, 18, 28]. It is noteworthy that this field of research has no available dataset that can be used to evaluate the approaches, or even to demonstrate their outcomes' quality. Given these

circumstances we have identified a research methodology and a benchmarking process that can help us to achieve our aims. Our research process is depicted by the following Figure 2.

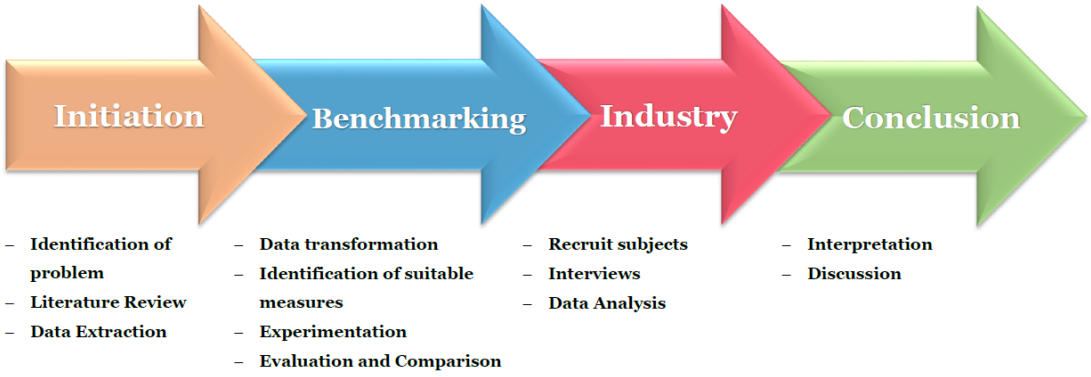

Figure 2: Research Methodology and Process

From Figure 2, it can be seen that the work carried out for this thesis consists of four stages. The first stage is the initiation, which encompasses three main activities. The first activity is reviewing the literature. The findings from the literature are presented in Chapter 2, which has helped to conclude the definition of SSSP problem. For this definition, suitable datasets are created by extracting information from historical software project records. These datasets are then transformed into different project scenarios by which different level of complexity are identified based on the level of project and resource's attributes presented in each. This transformation was the first activity in the second stage of the benchmarking. Benchmarking stage in addition to the data transformation includes identification of suitable performance measures for benchmarking the solution proposed for SSSP problem. The outcomes from both activities have helped to shape our benchmarking process presented in Chapter 3. Based on the benchmarking process, experimentation of a set of representative proposed solutions for SSSP problem are performed according to the identified levels of complexity. The outcomes from the experimentation are then used to evaluate, and compare these solutions against each other. These outcomes and findings from the evaluation and comparison are presented in Chapter 4 and Chapter 5.

On the other hand, an industrial exploration and evaluation stage is planned for this thesis depicted in Figure 2, and presented in Chapter 6. This stage starts by recruiting subjects from our industrial partners using a direct recruitment method. However, the main problem is finding a suitable time that can be agreed by all. Two phases are planned for this stage while interviewing subjects, where the aim of these phases is to explore the subjects' performance and the demographic information about their background and experience. Interpretations from the

benchmarking stage findings, and the interviews outcomes are then extracted enabling to conclude the research in Chapter 7.

1.6. List of Contributions

In pursuit of providing support for our work on this thesis, many experiments and systematic procedures were carried out. The main contribution of this thesis are as follow:

- A valid dataset that holds different complexity levels, as well as the optimal solution for each.
- A complete benchmarking and evaluation approach, combined with quality metrics, and accuracy measures.
- An evaluation and comparison of nine of the most referenced and cited SSSP approaches.
- Formalization and evaluation of four team allocation methods with consideration of project time minimization using genetic algorithm optimization and resource's productivity.
- An empirical evaluation of HRA aspects, and PMs' performance from different industrial settings.
- A research paper named Benchmarking and Comparison of Software Project Human Resources Allocation Optimization Approaches. A preliminary comparison and benchmarking study was performed and the outcomes of Five approaches of [14, 18, 20, 21, 28] are reported in this study paper. This work is reported in Chapter 4 and published in [29]. This paper is presented in 0Appendix B.

1.7. Thesis Structure

The reminder of this thesis is organized as follow. In Chapter 2, a thorough review of software project estimation and management techniques is carried out, including specific emphasis on optimization techniques (Search-based algorithms) and comparative studies of SPSPM approaches. In Chapter 3, our general methodology adopted and benchmarking approach are presented including a systematic process for categorizing and running experiments on SSSP approaches, as well as the datasets and quality measures to use for demonstrating the approaches' quality. Chapter 4 provides results of employing the benchmarking approach on nine SSSP approaches. In Chapter 5, results of advanced experiments are provided including optimization of four team allocation methods. Chapter 6 provides an empirical evaluation of PMs' practices and

solutions to HRA scenarios quoted from the datasets with accumulation on the findings from Chapter 4 and Chapter 5. In Chapter 7, we conclude this thesis by providing the overall picture of the findings from Chapter 4, Chapter 5, and Chapter 6, summarizing the contributions of this work, and discussing limitations and possible future directions.

Chapter 2 Literature Review

This chapter provides to the reader the current state of the art in the literature regarding the information that a software project and human resources can provide in Section 2.1, the optimization techniques (Search-Based Algorithms) that are applicable and suitable for SSSP problem in Section 2.2, and studies that compare and evaluate human resource allocation optimization approaches in software project management in Section 2.3. In addition, Section 2.4, provides details about the selected approaches for run-time benchmarking and comparison. Section 2.5 provides a background of the benchmarking processes proposed by different research papers, available datasets, measurements, and statistical tests that are available for SE research. Finally, this chapter ends with a discussion, and concludes the findings in Section 2.6.

2.1 Software Project Information

Software project provides many parameters and constraints that need to be taken into account while allocating resources. Many models for software projects have been proposed that capture these parameters and constraints, such as size, effort required, time, and cost.

2.1.1 Software Size, and Effort Estimation Models

Size is one of the variables that gives an indication about the amount of work that has to be done either by the workforces or the managers themselves. Different units can be used to estimate the size related to how the software will be developed. The most popular units used by many estimation approaches are Source Line Of Code (SLOC) and Function Point (FP) [30, 31]. The line of code represents the estimated number of code lines that a software product will have in actual development. Function point on the other hand describes the software in term of functions that should be implemented to achieve the customer requirement. In addition, both units are used by effort estimation models to predict how many resources the project needs for the actual implementation. Accordingly, software size is one of the basic variable for effort estimation. However, in early development stages accurate estimation is hard to achieve. It depends upon

decomposing and splitting the project into small pieces to gain an understanding of the abstract level of the business problem. Therefore, having the abstract view along with the details can help to predict the project size, and to estimate the effort required.

Two types of approaches can be identified for software project estimation which are judgement-based and model-based approaches. Judgement-based approaches rely on expert project managers' intuition to predict project size, productivity of developers, and estimate the effort. Model-based approach on the other hand uses mathematical equations that model the attributes of projects and developers to estimate the effort, time and budget of a software project.

As one of the model-based approaches, Function Point (FP) estimation model, introduced by [32], has been developed using the Function Point (FP) software size unit. The benefit of using the FP estimation model as it can simply be used in early development stages while clarifying the size of the intended software to the users or customers. More advanced model of FP has been proposed by [30] to support the FP-based effort estimation according to classification of project size and complexity from 24 software applications developed by IBM DP service (IBM information system service) presented by [32]. By using this model, software firms are able to demonstrate the intended work in the early development stages. This has made the estimation model widely accepted and also suggested by the International Function Point Users Group [33] for industrial use. However, the elements of project complexity are established based on specific programming languages and regardless of the productivity variation of resources. Moreover, this estimation model does not provide the allocation of resources that can best achieve the estimated effort and time as it depends on specific attributes and features that the project can provide.

The COnstructive COst MOdel (COCOMO) proposed by [34], is arguably the most well-known and most widely used model-based approach for software project cost estimation. The first version of COCOMO was proposed in 1981 and focussed on supporting the development of embedded software system, and was aimed in particular at a waterfall-based development methodology. COCOMO performs cost estimation for software development by modelling the size of the project in term of Kilo Line Of Code (KLOC). Productivity of resources on the other hand is expressed by means of the number of lines of code they can reliably produce during a given time interval. The cost of using a resource therefore directly correlates to the amount of time required to perform a project and the resources' salary. To estimate the time required to complete a particular project, COCOMO introduces Person-Month as an effort estimation unit. The actual value of the Person-Month estimation for a project depends not only on the size of a project, but also on the development team size assigned to project.

An advanced level of COCOMO is then proposed by [35], and named *Ada COCOMO*. This model, in addition to the elements considered by the previous model, takes into account module's structure and phases for development, as well as the Ada programming concepts. It uses the same equations and the cost drivers as well, but it introduces phases for the size estimation of projects regarding incremental development. Ada COCOMO uses function points to express the software size as it is easier to use in early development stages when limited information is available. Once the lines of code can be estimated with reasonable accuracy, the advanced stage switches to use this metric embedding with four exponent scaling factors that determines the project size.

To address the advancement of incremental, spiral, and object-oriented software development methods on estimation models, a new version named COCOMOII was introduced by [36] with changes made on the cost drivers, size estimation as well as the equations of COCOMO81 and Ada. These changes have introduced new cost drivers (now called Effort Multipliers (EM)) grouped into four different categories. COCOMO II replaces the mode of development in the estimation equation with five scale factors based on the Software Engineering Institute (SEI) process maturity factors and according to [37]. In addition, it takes into account the economies and diseconomies of scale on project size discussed by [38]. This is as how "Software cost estimation models often have an exponential factor to account for the relative economies or diseconomies of scale encountered as a software project increases its size." (p. 77) [36]. However, the model should be calibrated by the company's data to represent the local productivity according to [39].

The problem is that this model does not take into account the modern development methods and it even assumes that the workforces all are at the same level of productivity and expertise. Additionally, the software firms that use COCOMO have to calibrate these parameters and constant values according to their productivity and project historical data. Because of this, the usefulness of this model has recently been debated and some studies shows that the majority of project managers prefer expert opinions over mathematical ones [40].

In addition to the COCOMO and FP models, the Work Break down Structure (WBS), as one of the judgement-based model, can be used as a tool to support the PMs in decomposing and splitting their projects into manageable parts. Back in the 1950's, WBS has been developed by United States Department of Defence (DoD) to support military purposes [41]. After that WBS became useful in most of the 1960's projects in USA. In 1987, WBS became widely available to researchers and was used to support managers on their project work worldwide [42].

WBS supports the management process starting from planning, to execution, and then to reporting and controlling. So it can be used as a progress report mechanism to monitor the work

against the planned [43].The main idea of WBS is to start by defining the project scope. This can be done by defining the work elements. These elements can be represented as components of the original product, or activities of production towards the final product.

The elements that have been described earlier should be counted as deliverables. These deliverables have to be definable, manageable, estimateable, and measurable. This can lead the managers to estimate the whole project according to each deliverable. This work needs the managers and the experts in the field to use their expertise to define how long and how much each deliverable will cost.

Moreover, Delphi technique is another tool that can be used for effort estimation. This technique was developed in the 1940's and then published by [44]. Delphi technique is a judgement-based approach that has been successfully applied by many researches and for several purposes including software projects. It depends on a group of experts rather than the judgment of one expert. It includes four characteristics which are the anonymity of participants, information gathering type, feedback, and facilitator.

The facilitator starts by providing the questionnaire to each participant, monitoring the process, and recording the responses. His/her responsibility is to ensure that the anonymity rules the steps and that no participant will know who the others are and what are their responses. According to the defined reviewing times, there will be more than one stage for reviewing the experts' opinions. Feedback in each stage therefore can be made for the participants to review and update their responses. The participants are then should be all agree on the best of the responses. This technique provides a very useful tool for the researchers as well as the decision makers to forecast using a group of experts' knowledge. However, the outcomes of using it depends on the participants' knowledge, which might not conclude an optimal result.

2.1.2 Software Project Task Dependency Modelling

To capture a relation between several tasks, mathematical structures are used to model these into graphs. Two main graphs have been proposed depending on how the objects are connected with each other, which are directed graph, and undirected graph [45]. In software projects, the need is to capture which task should start before the other(s). Accordingly, the graph that can be used to demonstrate the dependency constraint between software project tasks is the directed graph.

One of the important application of Directed Acyclic Graph (DAG) is for project scheduling. In software project management DAG is used as a structure to show the Task Precedence Graph (TPG) presenting the schedule plan and the execution process of project [14]. However, it is the

responsibility of a project manager to ensure that the schedule is formed by a manageable timeframe and serves as a timetable for the project. Many techniques have been proposed as an application of DAG to depict the project schedule and to facilitate the project manager's work.

The Program Evaluation and Review Technique (PERT) is one of the application of DAG used as a project management technique for analysing and representing the completion time of project tasks introduced in the 1959 by [46]. This technique has been suggested to be combined with WBS for project and task time estimation. This technique has proposed a solution for two main aspects. The first aspect that this technique is concerned about is the schedule graph representation of project. The second aspect is the estimation of project task time.

For schedule representation, DAG is used to depict the project task workflow however, the representation of arcs and nodes are changed. A node in PERT represents a milestones to be achieved rather than specific task to be performed, and an arc represents the beginning and completion of a task and its estimated time. This time estimation requires several experts' opinions regarding the size and the time for each part of the project. The following Equation 9 developed by [46] for time estimation.

$$Estimated\ Time = (Optimistic + 4\ \times Most\ likely + pessimistic)\ \div 6 \tag{9}$$

The average estimated time for each project task in Equation 9 is calculated by the following attributes: 1) the optimistic value, which is defined supposing that the project will proceed better than expected, 2) the most likely value, which is the normal case happens, and 3) the pessimistic value which takes into account the worst and that the project development will consume more time than the expected. While this technique is powerful in demonstrating project schedules, providing the alternative project task workflow, and providing time estimate tool however, it has no precise measures in determining the values of the three attributes that can be counted as a weakness.

Another application of DAG is the Critical Path Method (CPM). This method is introduced by [47], and propose a mathematical method to define the longest path of time for a sequential series of tasks that contains dependencies between some or all of these tasks. The first purpose of this method was to facilitate the planning and scheduling of business management. This method presents the tasks or the activities by a graphical arrow diagram. However, nowadays this method can be used within the Gantt chart.

The benefit of using the CPM is accurately managing the efforts to the estimated delivery time by a single master plan. Besides, this method depicts the heart and the hard project tasks or activities

that the manager should carefully manage. The CPM process starts with the planning, which is defining the project tasks. Then scheduling the planned tasks by defining the required time for each task as well as the early and late time. By identifying these variables CPM can be then used to calculate the longest path among the alternative Path(s) that will help to determine the project duration.

The graphical arrow diagram developed by [47] shows the events, the jobs, and the series. The Events which refers to the products is represented by a circles in this diagram, where the jobs that done by a resource is represented by arrows linking between the events depicted in Figure 3. Three types of relation which are precedes, follow, and concurrent are between the events. Through these relation a well-defined order to perform the jobs is called then the series. However, two concepts are used in CPM to represent the relation, the origin that precedes, or terminus that follow another event.

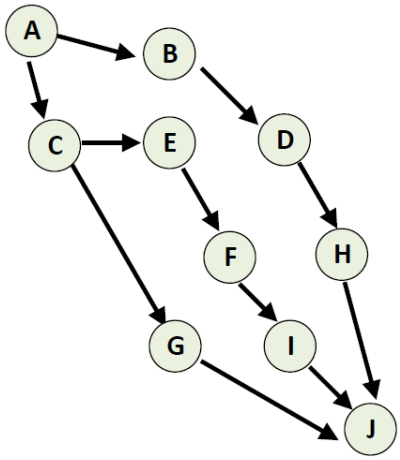

Figure 3: Critical Path Diagram

Three kinds of event's time in CPM are important to be defined for each event, which are earliest start time, latest completion time, and the job duration between two events. Based on these three variables criticality of an event can be calculated by measuring the differences between the early time, late time, and duration. The value of this difference is called floating time. If this value is greater than zero, then this event is called floater, or otherwise if it equal zero then this event is a critical one. The definition of project completion time by [47] is the late time of last critical event.

This method is one of the bases of current project management planning along with the Gantt chart. However, this method is heavily dependent on the PM to estimate the time variables for each event, and does not consider resource allocation while estimating the events' durations.

Another task management technique similar to DAG is the Gantt chart proposed by [48]. It graphically displays the order and dependencies of tasks in a diagram. The main aim by [48] was to provide balancing charts and machine loading of what the resources should do, and did do [49]. Gantt chart contains the project resources, and bars that represent the number of days for each resource in calendar that shows when the task of each resource will start and finish.

The benefits of Gantt chart is to represent the work plan as a progress report and graphical schedule [49]. It enables the manger to keep attention on overcoming obstacles and avoiding delays [50]. To clarify the idea of how we can draw a Gantt chart, the following example of three workers that they supposed to work in developing simple software project can be drawn as follow:

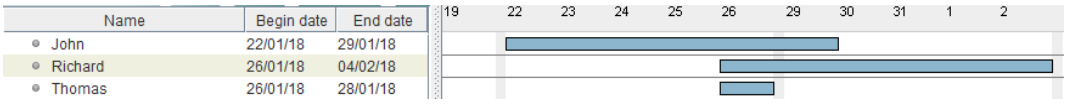

Figure 4: Sample of Gantt chart

As can easily be seen from Figure 4 that this project will start on 22nd of Jan and to complete in 4th of Feb, which takes two weeks. However, from this figure, it can be also seen that while John is working from 22nd till 26th of Jan, both Richard and Thomas are available for any other work to perform, and exactly the same for both John and Thomas after 30th of Jan. Diagramming using a Gantt chart has been one of the important tasks that the managers have to carry out. This chart provides very useful information for resource balancing, however it does not provide a mechanism for project task scheduling and optimal resource allocation.

On the other hand, to illustrate the availability of human resources in efficient sequence and to ensure the quality of project schedule, the Critical Resource Diagram (CRD) was introduced in [51] as a tool for managing the project workflow in term of resources assigned, rather than project activities [52]. Scheduling the resources by the CRD is similar to the arrow diagram presented by the CPM, except that instead of presenting the event's name in each circle, we need to include both the activity and the resource's name on it. This way of presentation is to show that this resource is unavailable during the activity time. In addition, through this diagram managers can identify any time conflicts between the resources and who are available for next activities.

The following Figure 5 depicts a CRD that shows five activities and their assigned resources within a specific sequence. This diagram can provide to managers whether any resource is involved in simultaneous activities, and to ensure that the resources are sufficient for the project activities.

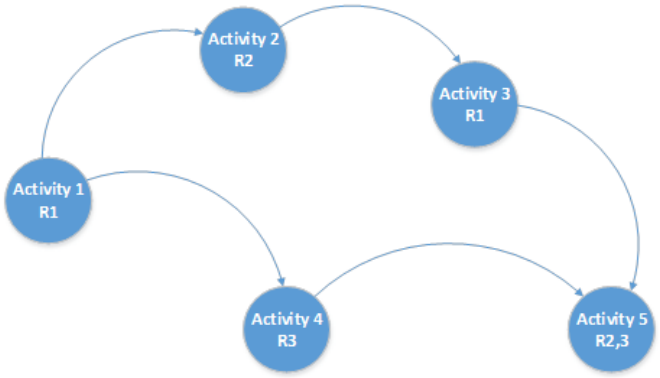

Figure 5: Critical Resource Diagram

CRD is a very useful tool that provides important information for managers to take into their consideration while distributing the resource to activities and balancing the resources loads, however finding a good quality resource allocation while using this diagram in only depending on the manager's intuition and expertise.

2.1.3 Workforce Models

In software projects competencies are counted as a key for productivity measurement. This productivity can leads to efficient time plans, and high quality products. In addition, human resource competencies are the essential input for a successful allocation in software projects. Modelling human resource competencies, therefore, has emerged as one of the important aspects in human resource allocation.

The results of a study presented in [53] classifies workforce competency models into three categories. These categories are supporting the performance of individuals, groups (collective), or organizations (global). The individual models category takes into account the models that are related only to the technical capability of human resources. The collective models category on the other hand take into account the models that describe the competencies of team roles such as analyst, designer, programmer, etc. The final Global models category takes into account the models that created for the organization's future so that dynamic improvements are incrementally made on the overall performance.

However, based on how the competencies will be used by the allocation approaches the workforce models are divided in this section into qualitative and quantitative models. Qualitative models are the models that use the competencies with binary representation to show the existence of a quality or not. Quantitative models on the other hand aim to quantify the workforce attributes for mathematical use. The following sections cover the current state of the art of each model type.

2.1.3.1 Qualitative Workforce Models

US department of labour IT competencies INFOCOMP (2012)

The model presented in [54] aims to clarify the competencies of workers required in the information technology and software development industry. Competencies presented in this model are grouped into four tiers. Tier one contains the IT competencies counted as a personal effectiveness. Tier two describes the competencies that should be established during academic life. Tier three considers the competencies that the resource should gain from the workplace. Tier four describes the industrial technical competencies.

From Table 2, we can see that these levels are presented as building blocks. Accordingly one of the aims of this competency model is to evaluate the IT workers. The details of the tiers and the competencies introduced by the INFOCOMP are listed in the following table:

Table 2: Attributes of INFOCOMP 2012 [54]

Tiers	Competencies
1. Personal effectiveness	Interpersonal skills and team work
	Integrity
	Professionalism
	Ethics
	Adaptability & flexibility
	Dependability & reliability
	Lifelong learning
2. Academic competencies	Reading
	Writing
	Mathematics
	Science
	Communication: Listening & Speaking
	Critical and Analytic Thinking
	Basic Computer Skills
3. Workplace Competencies	Collaboration
	Planning & Organizing
	Innovative Thinking
	Problem Solving & Decision Making
	Working with Tools & Technology
	Business Fundamentals
4. Industry-Wide Technical Competencies	Principles of Information Technology
	Information Management
	Networks & Mobility
	Software Development
	User & Customer Support
	Digital Media
	Compliance
	Security & Data Integrity

The competencies presented in Table 2 can be used to assess the level that a software organization has according to their workforce's available competencies. The competencies presented in tier one along with tier two are the basic level of competencies for the workforce to start in the IT and software development career. The workforce then during his/her career have develop his/her competencies to tier three, and four. In addition, this can be used as a future plan for individuals to improve their skills and competencies. However, this model is proposed as the basis of a benchmark for gap analysis that the industry and academic institutions have to use to comprehend the quality of their individual workers.

IEEE Software Engineering Competency Model (SWECOM)

The IEEE Software Engineering Competency Model [55] aims to improve software industry workers' capabilities. In addition, it can be used to assess the current outcomes of educational bodies as explained in [56]. Based on differentiating between knowledge and skills, SWECOM represents knowledge as an element for establishing a good skill. The difference is that the knowledge is what the individual knows and skills are what the individual can do.

Moreover, this model is not only proposed for software engineers, other related disciplines are considered as well. The related disciplines that are mentioned include computer engineering, computer science, general management, mathematics, project management, quality management, and system engineering. These disciplines are required in software projects and accordingly this makes them count as another element of SWECOM.

SWECOM contains five elements that establish a foundation for the workers in the software industry. These element as can be seen in Figure 6 are behavioural attributes and skills, related disciplines, requisite knowledge, cognitive skills, and technical skills.

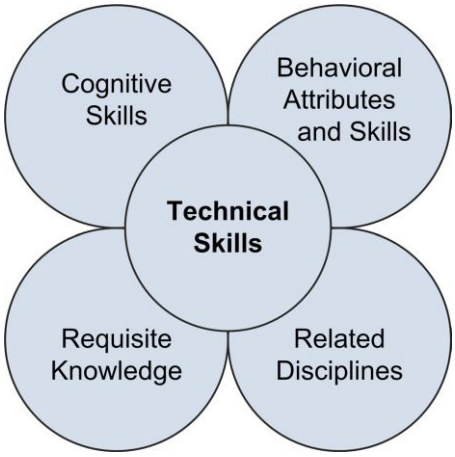

Figure 6: SWECOM Elements [55]

The considered Cognitive skills are reasoning, analytical skills, problem solving, and innovation. Behavioural attributes and skills on the other hand are: aptitude, initiative, enthusiasm, work ethic, willingness, trustworthiness, cultural sensitivity, communication skills, team participation, and technical leadership skills.

The most important part specially for measuring the fitness of the individual in this model is the technical skills which are categorized based on the phases of development and crosscutting of the different disciplines related to software. This model has in addition classify the level of involvement of the individual software engineer in each project activity into five, which are follows, assists, participates, leads, creates. Moreover, this model provides classification of the individual competency skill level that expresses how the individuals would fit to the work, which are technician, entry level practitioner, practitioner, technical leader, senior software engineer.

This model provides gap analysis worksheets that can be used to assess both individuals, and project team's competencies. These worksheets demonstrate the gap by using the activity and competency levels to fill the current and the needed skills. This would give an example of how this model can be used to measure the fitness of worker to a specific requirement either for the firm, project, or individual level.

This model provides a tool that can be useful for assessing the fitness of resources to projects. However, resource allocation does not only requires staffing, but also to consider the inter-dependency nature between project tasks.

Psychological Capability of Human Resource

A different side of software human resources rather the technical capability is the psychological aspect. In [57] a method is proposed for assigning workforces to development roles based on their psychological capabilities. These capabilities and factors are addressed by psychologists and software project managers using standards and frameworks. The standards and frameworks used in [57] are the Assessment Centre Method (ACM) framework [58] and the 16PF personality factors psychological tests [59].

The ACM framework offers a process for selecting the best suited for a job containing individual characterization, identification of roles capabilities, and matching individuals to roles. The ACM framework requires evaluators (psychologist) to weigh and categorize the psychological factors into several capabilities domains [58, 60]. The 16PF personality factor test on the other hand provides a questionnaire to evaluate the individuals in term of personality (psychological) factors

[59]. However, ACM has been used for the model presented in [57] as a base for validating the capabilities identified by 16PF test using psychologists.

In addition to the 16FP, the study presented in [57] uses an additional five personality dimensions addressed by [61]. The model presented in [57] is a binary evaluation that describes whether the individuals have these capabilities or not. The result of their work of a relational table of personality factors and the capabilities based on their study of real software organizations shows that each team role should have a specific capabilities presented in Table 3:

The major findings of this study was that 1) The defects rate decreased for 47% in projects that used this model to assign the resources, 2) Mean effort deviation reduced for about 30% and, 3) Ratio between estimated function point and actual effort improved to 44%. The reason for these improvements mentioned in [57] is that the resources of the sample organizations that used the model were more motivated since their personality factors had been considered.

Table 3: Team Roles and Personality Factors Relation [57]

	Intrapersonal												**organizational**			**interpersonal**			**Management**		
Software Roles	Analysis	Decision-making	Independence	Innovation and creativity	Judgment	Tenacity	Stress tolerance	Self-organization	Risk management	Environmental knowledge	Discipline	Environmental orientation	Customer service	Negotiation skills	Empathy	Sociability	Teamwork and cooperation	Co-worker evaluation	Group leadership	Planning and organization
Team leader	✓	✓		✓			✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
Quality manager	✓	✓	✓	✓			✓	✓				✓			✓		✓	✓	✓	✓
Requirement engineer	✓				✓		✓					✓	✓		✓	✓	✓			
Designer	✓	✓	✓			✓	✓	✓			✓	✓			✓		✓			
Programmer	✓	✓	✓			✓	✓	✓			✓	✓			✓		✓			
Maintenance and support specialist						✓	✓	✓			✓	✓	✓		✓		✓			
Tester			✓		✓		✓	✓			✓	✓			✓		✓			
Configration manager			✓		✓		✓	✓			✓	✓			✓		✓			

Psychological aspects of software resources have never been addressed before but it is important to evaluate, especially at team formation time. However, the model ignores the technical skills and the interdependency nature of software project tasks which leaves it with limited applicability. Additionally, sharing resources, as is the current practice with software firms, has not been addressed by this method.

2.1.3.2 Quantitative Workforce Models

Up to now we have seen qualitative resource models describe capabilities using yes no mechanism. This section presents a different type of resource model that uses a numerical level of importance scale for each competency that workers have to different development domains. This section accordingly covers quantitative models that have been proposed to represent the workforce capabilities.

Team Oriented Competency Model

The approach described in [62] is one of the collective competency models as its name implies dealing with formation and building the teams (team oriented). It is developed as a numerical model based on [57] described in the earlier section. Competencies proposed by this model have been verified by a group of software project managers. The process of verifying the model was adapted through two stages of the Delphi technique [44]. The resulted competencies are then correlated to team roles. Team roles however are identified in this method based on the Rational Unified Process (RUP) and Team Software Process (TSP) methodologies presented in Table 4.

The value presented in Table 4 represents the relation between the roles and the competencies defined by [62]. These values express the level of importance for each competency versus team role. For more information, please see page six of [62].The values and their expression are as follow:

1. 0 means that this competency is irrelevant for this role.
2. 1 indicates that this competency is fairly necessary.
3. 2 means that this competency is critical.
4. 0.5 represents that this competency has no agreement by the participants of [62] study, and
5. 1.5 considered necessary but not by all the participants.

This approach uses the Myers-Briggs Type Indicator (MBTI) and Belbin team inventory psychological tests to verify the workforce's capabilities. MBTI as presented in [63] assesses the personality type of workers. It contains four categories where each one has a pair of factors to be assessed for each worker. These pairs are Extroversion and Introversion, Intuition and Sensing, Thinking and Feeling, and Judgement and Perception.

Table 4: Team Roles and Competencies [62]

Software Roles		Teamwork and cooperation	Analytical skills	Planning and organization	Capacity to control	Negotiation skills	Adaptability (Flexibility)	Research skills	Strategy skills	Conceptual thinking	Risk management	Decision-making	Tenacity	Independence	Customer service	Commitment to the organization	Proactivity	Oral communication skills	Written communication skills	Continuous Learning Ability	Language proficiency	Programming Language proficiency	Modelling tool proficiency	Database management system proficiency	Software development methodology proficiency	Graphic design abilities	Product knowledge
		General competences																			Technical competences						
Project leader		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	1	2	2
Graphic designer		2	2	1	0	1	2	2	0	1	1	1	1	2	2	2	1	1	1	2	2	0.5	0	0	0	2	2
Architect		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2
Analyst		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	1	2
Designer		2	2	2	2	1.5	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	1	2
Database Designer		2	2	1	1	1	2	2	1	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	0	2
Programmer		2	2	1	1	1	1	2	1	2	2	1	2	2	1	2	2	1	1	2	1	2	2	2	2	1	2
Tester		2	1	1	1	0	1	1	1	1	1.5	0.5	2	1	2	2	1	1	2	2	1	0	1	0	0	0	2

Based on those tests, rules for the assignment was established to assure that the workforce(s) fit to the role. For instance, MBTI is used to justify if the workforce is fit to be a project leader. Their study shows that project leader should have Extroversion and Judgment capability of MBTI dimensions in order to be capable to lead the project.

The Belbin test, on the other hand, as described in [64] contains three role types, Action, Mental, and Social roles. Two important capabilities of these roles, the Shaper, and Chairman are used by [62] to justify the project leader capabilities. Another rule is that the overall preferences of team

Belbin roles should be for action role over mental role, and mental role over social role. In addition, the study shows that at least one of the team member should have plant role in order to increase the team performance.

In this model we can see the additional technical capabilities over the one presented in [57] that have been taken into account, as well as quantifying these capabilities by [62]. However this model does not take into account the interdependency nature of software projects and developers sharing that software firms do nowadays.

Best-Fitted Resource Model

The Best-Fitted Resource (BFR) methodology proposed in [7] takes into account the learning ability of resources of technical skills. The methodology uses the relationship ability matrix for all the skills available by resources and those required for a project. The relation ability matrix expresses how a group of skills can impact another.

BFR methodology suggests seven criterion to be used with the skills matrices. The value for each one of these criteria ranges from 0 to 1. These criteria and their notations are the following:

- The expected use of skill j on task t ($e_{j,t}$).
- The complexity of skill j on task t ($c_{j,t}$).
- The significance of skill j on task t ($s_{j,t}$).
- The relation between the knowledge of skill j and skill k ($r_{j,k}$).
- The knowledge level of resource y to skill j ($l_{y,j}$).
- The relation between the set of skills of resource y and the required skill j ($b_{y,j}$).
- The fitness of resource y to task t ($f_{y,t}$).

BFR contains four steps, each one results in a table. The first one is concerned with task required skills (TRS). The second one is concerned with skill relationship (SR). The third is concerned with resource's skillset (RSS). The last one produces the best-fitted resource to each task (BFR).

In the first step, the value of $e_{j,t}$ and $c_{j,t}$ for required skills are estimated. Values of expected use of skill are (0.3) which means little use, (0.7) means significance use, and (1.0) means extensive use. Complexity of skill, on the other hand, has values of (0.2) that means simple, (0.5) means complex, and (1.0) very challenging. The significance of skills $s_{j,t}$ is then calculated as the product of both. This results in table containing the skills, the use, the complexity and the significance of each as the following example in Table 5:

Table 5: Example of Use, Complexity, and significance of skills

Required skills	$e_{j,t}$	$c_{j,t}$	$s_{j,t}$
Hardware	0.3	0.5	= 0.15
PHP	0.7	0.2	= 0.14
.Net	1	1	= 1

The second step is concerned with the relation between skills. The result of this step shows the learning ability regarding the relation between these skills. This relation can be defined using the criteria of $r_{j,k}$. The values of skill relation (0) which means no relation, (0.2) weak, (0.5) intermediate, and (1.0) as strong relation. Using the skills from previous example will be as the following Table 6:

Table 6: Skills Relation

Skills relation	Hardware	PHP	.Net	Java
Hardware	1	0	0	0
PHP	0	1	0.5	0.5
.Net	0	0.5	1	0.2
Java	0	0.5	0.2	1

The third step involve the resources' skills set (RSS). The project manager in this step ranks each resource for each skill as the value of ($l_{y,j}$). Values considered are (0) for no knowledge of the resource in this skill, (0.2) low knowledge, (0.5) intermediate, and (1.0) is high. Accordingly this can be demonstrated for our example by the following Table 7:

Table 7: Resources Knowledge level for each Skill

Resources' skills	Hardware	PHP	.Net	Java
Resource 1	1	0.5	0	0.5
Resource 2	0	1	0.5	0.2
Resource 3	0.5	0	0.2	1

The fourth step considers training time of each resource. The relation between skills indicates whether the resource who possesses a certain skill can develop himself within a short time for a related one. This step can be achieved by using the results of steps one and two and combining them into one table. The factor of ($b_{y,j}$) at this step shows how the resources are fitted to the task

by considering each value of, required skill (k), resource skills, as well as the relation between the skills, then it can be calculated by:

$$B_{yk} = \max_{h \in H}[l_{yh} * r_{hk}] \quad (10)$$

The following table shows how this step can be obtained.

Table 8: Resources Fitness to Projects

Resources And their skills	**Skills Required**				**Fit**
Resource 1	**Hardware**	**PHP**	**.Net**	**Java**	$b_{y,j}$
Hardware	1	0	0	0	1
PHP	0	0.5	0	0.25	0.5
.Net	0	0.25	0	0.1	0.25
Java	0	0.25	0	0.5	0.5
Resource 2	Hardware	PHP	.Net	Java	$\boldsymbol{b_{y,j}}$
Hardware	0	0	0	0	0
PHP	0	1	0.25	0.1	1
.Net	0	0.5	0.5	0.04	0.5
Java	0	0.5	0.1	0.2	0.5
Resource 3	Hardware	PHP	.Net	Java	$\boldsymbol{b_{y,j}}$
Hardware	0.5	0	0	0	0.5
PHP	0	0	0.1	0.5	0.5
.Net	0	0	0.2	0.2	0.2
Java	0	0	0.1	1	1

We can see from Table 8 that if we defined the relation between the skills, then we can see that if the resource do not currently have a skill then (s)he could after a short training be able to acquire the required skill(s). This can be seen by the resource 3 that his earlier knowledge of PHP was (0), however through the relation between PHP, Java, and .Net, his/her skills can improve specially for PHP by limited time of training so he can be able to do the task.

As this method can efficiently demonstrate in a quantitative manner the availability of skills required for project tasks among the available resources, however it does not provide a mechanism for resource allocation and project task dependency handling.

Other Human Resource Attributes and Capabilities Models

The formal assessments used by the software industry and researcher focus on effort and productivity of developers. Many attributes can lead to understand the performance and

productivity of developers. Such attributes can be obtained by observing behavioural patterns of developers. One of the tools that enables the researchers to observe the developers' behavioural pattern is the Version Control System (VCS).

The study presented in [65] based on observation and pattern verification of developers through VCS shows that three attributes would contribute to productivity measurement and developer assessment. Takeover, which is the first attribute, indicates when the developer writes codes in a short period of time. Bug Fix on the other hand, indicates the amount of corrections made on the developer's code. The third is Teamwork that leads to an understanding of the collaboration between team members.

Another model presented in [66] introduces two metrics that can be estimated using VCS. These metrics are the effort (Productivity) and code-survival of a single developer. Productivity in this model is estimated based on how many files the developer can produce in the VCS during a unit of time. The code-survival metric on the other hand, is the amount of code of a developer that has never been changed by anyone.

Effort and Code-Survival assessment metrics in [66] are estimated based on three development operations. The first operation is Add, which means adding new code to the development file. The second operation is Modification, which means in case of modifying any existing code of the development file. The final one is Deletion of code or file of the developer.

An alternative workforce model presented in [16] approach takes into account technical skills grouped into three categories. The skills presented in these categories are not limited to those skills, and the model as mentioned in [16] is open to any new skill. The skills of the first category includes relationship with people, negotiation, and team work. The second one includes requirement elicitation, object-oriented analysis, databases, object-oriented design, Java, and test techniques. The final one includes just one characteristic which is experience in telecommunication.

Each one of the characteristics presented in [16] is ranked by a numeric value. The ranking value of first group varies from 1 to 3. Value of (1) means that the developer was trained on the subject, value of (2) means that he has ability, Value of (3) means he has great ability. The second group varies from 1 to 3, but value of (1) means knows and can perform under supervision, value of (2) means knows and can perform without supervision and Value of (3) means that the developer is an expert. Group three is different in the range. It starts from 1 and the highest is 4. Value of (1) for this group means that the developer has experience of between 2 and 6 months, Value of (2)

means his/her experience is between 6 months and 1 year, Value of (3) means his/her experience is between 1 and 3 years, and Value of (4) means that his experience is more than 3 years.

2.1.4 Discussion on software project information

Throughout the earlier presented estimation models, we can see that the effort is a fundamental factor and the basic parameter used to estimate the time and cost of a software project. Moreover, software effort estimation is modelled based on five factors as being described by [39]. These factors are identified by [67] as personnel (Developers), product size, development process, required product quality, and development environment. However, these factors are introduced in the estimation models as constants based on studies of historical data of productivities, qualities, schedules and/or processes gathered from real software projects.

Many researches have explored the differences between model-based and judgment-based estimation models as in [40, 68-70]. However, the organization sample studied in [40] reveals that most software organizations use judgement-based approach as they are concerned about the accuracy of model-based approaches. In addition, in [71] stated that no model or method of effort and cost estimation is better than another. Likewise, we do not know yet how to accurately estimate the effort for mega-large software projects, to measure size and complexity accurately for software, and to predict team's and individual's productivity.

It is noteworthy that the approaches that optimize for software project management issues such as [21, 22] consider the effort as a valid input to the approach, and the effort in these approaches is mainly measured in terms of Man-Month, or Man-Days using COCOMO models.

In addition to effort estimation, project task dependency and scheduling has received the most research attention during the last century for project scheduling. Amongst the different techniques that have been proposed to depict the precedence relationship and dependency between the project tasks, TPG depicted by DAG is the main technique used by the optimization approaches for this matter.

Workforce models, on the other hand, are the part of project management information neglected by SSSP approaches as in [14, 15, 28]. It is understandable that the optimization process needs approaches that provide quantitative attributes over the qualitative ones. However, few of SSSP approaches provide list of skills and competencies of software project development as in [28] and [23] ones. These approaches have provided the roles that a software developer might have, and those required for a project task to be performed. Workforce models can work as a supporting mechanism for resource skill constraint handling during the optimization process including the

fitness function. A binary skill selection within many of SSSP approaches' optimization process has been used as in [14, 18]. However, this type of binary selection without consideration of productivity can fail the search to find a good skilled resource for the resource allocation problem within a reasonable computation time. For that reason, there are some approaches that have included productivity in their optimization problem and relaxing the optimization constraint to a better productive resource who are available at the time of need. In that sense, models as best-fitted resource model can be more applicable in resource allocation optimization.

2.2 Optimization Techniques (Search-Based Algorithms)

Optimization is a branch in mathematics that focuses on techniques able to find an optimal or near optimal solution for a given optimization problem. An optimization problem can be represented as a problem of minimizing or maximizing an objective or goal. To solve this problem, the optimization techniques employ a function within the search process to measure the fitness of the alternatives for the fastest, cheapest, lowest, etc, solution. This function can be categorized into two according to the optimization problem. If the problem is to search for the minimized solution, then this function is called a "Cost" function. If the problem is to search for the maximized solution, then this function is called a "Utility" function.

These functions search for the "fittest" solutions amongst the generated alternatives. However, according to the optimization problem there might be a constraint(s) that has to be applied to measure their feasibility. Two types of constraints can be used for an optimization problem. The first type is a "Soft" constraint. The second type is "Hard" constraint. If the violation of a constraint is considered as unfeasible solution, then this constraint is called hard. On the other hand, if the violation of a constraint will still be considered as a feasible solution but a penalty might be applied on that solution, then this constraint is called soft.

The optimization techniques are algorithms capable of searching for an optimal or near optimal solution(s), and in that sense they are called Search-Based Algorithms too. These techniques or algorithms are categorized into three groups; exact, heuristic, and Meta-Heuristic techniques.

Exact optimization techniques are techniques that form a branching and exhaustive search that guarantee finding the optimal solution, such as branch and bound, and branch and cut [72, 73]. However as the problem scales up, then exact techniques would not be beneficial because of the vast processing time needed to compute the optimal solution.

Alternatively, heuristic techniques can be used to determine, not perfectly accurate, but good quality approximations, such as Greedy algorithms, Hill Climbing (HC), and Dynamic

Programming (DP) [74]. However, these techniques tend to find a fast solution at the expense of memory for DP, or the solution quality for Greedy. The main drawbacks of algorithms belonging to this class is that the solution obtained might be trapped into a local optima.

On the other hand, Meta-Heuristics such as Genetic Algorithm (GA), Particle swarm optimization (PSO), etc. represent a class of generic optimization techniques using ideas from various fields as inspiration for the process of trying to solve optimization problems. These techniques are nature-inspired algorithms developed by mimicking the most successful selection, and behaviour processes in nature. These techniques have the power of learning throughout the search. That means while stochastically creating solutions, these solutions are compared heuristically so that either the least costly or most utilized result is searched. This however, comes at the cost of using machine memory. Therefore, metaheuristic techniques attempt to solve the problem by intelligently visiting only some solutions, but there is no guarantee that the best solution is returned [74].

The following subsections discuss the optimization techniques sorted first by the exact, the heuristic, and then meta-heuristic techniques.

2.2.1 Branch and Bound

Branch and bound is one of the best general technique for solving constrained optimization problems [72]. This technique intelligently structures the search space for all feasible solutions. Feasible solutions in Branch and bound are partitioned into smaller and smaller sub branches as a tree and a lower bound is calculated for the minimized solutions within each sub branch. In each partition, the bound of the sub branches that exceeds the minimum of a known feasible solution is excluded from all further partitions. Partitioning continues until a feasible solution is found such that its cost value is no greater than the bound for any sub branches.

The number of pruning of branches that occurs in branch and bound is large. Consequently, the algorithm is powerful, searching effectively within the feasible branches. However, obtaining an optimal solution using this technique requires ignoring the computational time matter specifically for large-scale problems. For more details, see [72, 75, 76].

2.2.2 Backtracking

Backtracking is a technique that can be used to find a partial, or all solutions, to a constraint satisfaction problem [77]. This technique is usually combined with an optimization algorithm to incrementally build candidate solutions by determining and abandoning the partial candidates

whose solution cannot be successfully completed. This technique can be applied when the problem accepts the concept of near optimal solutions. The technique enumerates a set of partial solutions. With extension steps, candidates are determined incrementally to complete the whole set. The partial candidates by this technique can be seen as nodes of tree. Backtracking, as its name implies, is the search that is done recursively in finding solutions starting at the root to the end of a branch. With a given criteria, the best partial solution can be obtained. However, if the problem is large in term of number of variables consequently the search will consume more computational time. For more details about this technique see [77].

2.2.3 Branch and Cut

Branch and cut is a method for solving an optimization problem restricted to integer values. This method involves the branch and bound technique within the search where a cutting plan is used to constrict the relaxation of the problem. While solving the relaxed problem, and not being successful in pruning the node on the basis of the constrained solution, the search tries to find a violated cut. The violation cut will hold a solution that do not satisfy the constraint. If one or more violated cuts are found, they are added to the formulation and the problem is solved again. If none are found, then the method branches again. This technique suffers from processing time issues as in the branch and bound, and many decisions have to be made regarding the strategies for branching on a variable. For more details about this technique see [76, 78].

2.2.4 Greedy

The greedy algorithm is one of the simplest algorithms that can be used in optimization, however, there is no guarantee that the solution output is an optimal one. Greedy algorithm is often used to solve optimization problems that either maximize or minimize an objective with a set of constraints. Greedy algorithm can be seen as a process that starts from an initial node of the problem and goes to the last node. The algorithm starts with the initial node to search for an optimal solution to it. As the algorithm progresses with problem nodes, choices for better solutions become fewer for further nodes. The final solution by this method could fall into a local optima rather than to go for a global optima within the search space. For more details about this technique see [79, 80].

2.2.5 Dynamic Programming

Dynamic programming (DP) is a useful mathematical technique for making a sequence of interrelated decisions and solving a complex problem by breaking it down into a collection of

simpler sub problems. There does not exist a standard mathematical formulation of the dynamic programming problem. Rather, dynamic programming approaches a problem by identifying a set of choices to be used to fit the problem's decisions. Dynamic programming solves the optimization problems recursively by decomposing solutions to the sub problems. When sub problems overlap, Dynamic programming solves these sub problems just once recursively and then combines their solutions to solve the original problem. Dynamic programming optimizes the solution by either searching for a minimized or maximized value. Moreover, Dynamic programming stores the answer avoiding rework on the same solutions. However, this method is considered memory consuming, thereby saving computation time at the expense of storage space. For more details about this technique see [80].

2.2.6 Hill Climbing

Hill climbing (HC) is a mathematical technique that can be used to solve an optimization problem by accepting a solution within the local optima [81]. The solution that hill climbing offers is found by a random search that proceeds from an initial point of the problem and searches for a best solution within the neighbours of that point. Once a better neighbour is found this becomes the current point in the search space and the process is repeated until no further improvements can be found. Here the search terminates and a maxima (highest point) has been found. The technique called hill climbing, because the search space for the objective to maximize can be seen as a topography that contains peaks "Hills" where the technique searches for the peak within the hill that contains the random point selected. This technique can be easily implemented, but it might struggle with a local optima within the solution space. So, the solution obtained by hill climbing could be far poorer than the global maxima –best solution within the search space. Hill climbing shows a robust and useful application in software engineering [10]. For more details about this technique see [81, 82].

2.2.7 Genetic Algorithm

Genetic Algorithm (GA) was originally proposed by John Henry Holland in [83]. As one of the meta-heuristic techniques, GA is one of the most popular, used and applied to the problem of Search-Based Software Engineering (SBSE) in more than 80% [5] of the approaches proposed so far. GA uses concepts of genetics, such as population and mutation to solve an optimization problem [84]. Metaheuristic in this technique is designated by two genetic operations crossover and mutation. A crossover operation creates solutions in which the structural information of two solutions are crossed to generate two new solutions [85]. On the other hand, mutation process do

random changes on the solutions generated. The mutation operation is used to avoid same solution generation, which can lead to explore various search spaces [85]. Solutions are evaluated to determine which will continue to the next iteration by continuous selection according to the objective function [74]. This technique can be used with a problem where finding a precise global optimum is less important than finding an acceptable solution in a fixed amount of time. For more details about this technique see [74, 84-86].

2.2.8 Multi-Objective Genetic Algorithm

Multi-objective Genetic Algorithm (MoGA) is an expanded genetic algorithm that handles more than one objective where these objectives cannot be combined into a single objective with a weighted scoring model. The idea for using multi objective rather than combining them is that these objectives are generally conflicting, preventing simultaneous optimization of each objective. In this technique, a number of solutions can be found so the decision maker will have an insight about the problem characteristics before making decision on the suited final solution. The solution to this problem is not a single point, but a family of points known as the Pareto-optimal set. This is due to the fact that most real engineering problems actually do have multiple-objectives, i.e., minimize cost, maximize performance, maximize reliability, etc. These are difficult but realistic problems [87].

An example of a MoGA is the algorithm Non-dominated Sorting Genetic Algorithm II (NSGAII) proposed in [88]. This algorithm uses a sorting approach that facilitates the search of GA and reduces the computation complexity. One of the MoGA downsides is that where there are of complicating factors the technique will consume more computational time. On the other hand, the user might have to define several options for different solutions. For more details about MoGA and NSGAII technique see [88, 89].

2.2.9 Simulated Annealing

Simulated Annealing (SA) is one of the metaheuristic techniques that approximates the objective function based on a physical process that occurs in metal's metallurgy. This technique is used to approximate a global optima within a large search space. The inspiration comes from the tempering process. This process aims to crystallize a material with minimal energy. The process starts by heating the material to high temperatures and, thereafter, is cooled so that at the end of the process the material is crystallized by a minimal energy. Here the tempering process in metal's metallurgy can be seen as a mathematical optimization. This optimization is to minimize an objective function such the one used for tempering process for energy minimization. The

algorithm of Simulated Annealing allows for solutions that will not improve the value of the function. Accordingly, the algorithm can overstep the search to find global optima.

The simulated Annealing algorithm accepts the solutions by two criteria. The first is a direct one which if the new solution is better than the current solution then it is accepted. The second criteria are based on probability. If the new solution worsens the objective, then it is accepted with a certain probability defined according to three aspects, the difference between the solutions, the current value of the variable temperature, and constant physical value.

This technique obviously works with a minimization problem. Accordingly, it becomes harder to be implemented to maximize an objective. This technique can be used with a problem of finding a precise global optimum is less important than finding acceptable solution in a fixed amount of time. For more details about this techniques see [74].

2.2.10 Particle Swarm

Particle swarm optimization (PSO) is a Meta-heuristic technique that can be used to solve an optimization problem by iteratively trying to improve a candidate solution based on a fitness function [90]. It is adopted from the observation of the natural behaviour of birds and fishes. This technique generates a population “Particles” and searches the solutions from the population according to position and velocity defined for particles. This technique formulates the search according to particles’ movement, which is updated by other particles for better positions and is expected to move the swarm to the best solution.

The method has shown very good performance on many benchmark problems while its rotation invariance and local convergence have been mathematically proven [91]. PSO can also be used on optimization problems that are partially irregular, noisy, change over time, etc. [92]. However, the choice of PSO parameters can have a large impact on optimization performance. Selecting PSO parameters that yield good performance has therefore been the subject of much research[90]. For more details about this technique see [90, 92].

2.2.11 Discussion on optimization techniques

Different optimization techniques are applicable to SSSP problem. However, these problems have been defined on many occasions as an NP-Hard problem [93], which leads us to the use of meta-heuristic techniques as the most useful in terms of computation time suitable for the hardness and complexity of this problem. It is important to notice too how the formulation of this problem can lead to different conclusions. A problem with a single person to a single task is a linear

programming that one can employ heuristic techniques to provide a solution. On the other hand, assigning multiple resources to project tasks, and scheduling these tasks according to the dependency relationship has a multiple stage solution that requires compromising the solution quality for fast computation using Meta-Heuristics.

From the next Section 2.3, we will see that Meta-Heuristics such as SA, and GA are the most widely used techniques in solving different SE problems. Both techniques have been employed by many approaches as in [20, 22] to provide evidence and benchmark the performance and suitability of these techniques to SE problems. In addition, multi-objective optimization for SSSP problem considering time, and cost, are proposed as in [14, 22] using GA. Part of the approaches that solve SSSP problem will be described and detailed in Chapter 4 as they will be subjects for our benchmarking and evaluation to SSSP approaches.

2.3 Comparative Studies in Optimization Approaches for SSSP Problem

This section reviews the current state of the art in the literature regarding studies that compare and evaluate optimization approaches for software project management problems. So far, only two studies have been published that compare and evaluate the optimization approaches of SSSP problem. The first study presented in [24] evaluates the proposed optimization approaches with the possibility of adoption within the software industry. The second study presented in [5] researches the optimization approaches to address future trends and promising areas of human resource allocation optimization. The observations and findings from these studies highlight categories of optimization approaches, the important attributes that these approaches adopt, and the approaches that are most useful in an industrial settings. Overall, each has placed an emphasis on possible future trends.

This section addresses the criteria used by both [5, 24] in Subsection 2.3.1, the observation and findings in these studies in Subsection 2.3.2, and summaries their findings in Subsection 2.6.

2.3.1 Criteria

By running an evaluation on fifty-two research papers, the study presented in [24] compares the SSSP approaches presented in these papers according to three criteria. These criteria are usefulness, work compatibility, and ease of use. Usefulness in this study is defined as the benefits that software firms might gain by adopting the solution proposed by an approach. Work compatibility is defined by the study as the fitness of proposed solution within the work

environment of software firms. Ease of use moreover, is defined by the study as how easily the approach can be adopted. In addition, the work presented in [24] discusses the aspects of problem concepts, development, and validation presented by the research papers included in their study. Problem concepts represent the optimization problem addressed by the approach. Development is the ability of the approach to integrate with a project management tool. Validation on the other hand, is the techniques used to validate each optimization approach.

The aim of the study presented in [24] is to identify aspects related to the difficulties in adopting the research papers' proposed solution by software organization. They have accordingly performed a systematic literature review to cover the concepts used within the research papers by extracting the texts that describes the problem model of the papers and categorising them. This has therefore enabled them to identify the relationships between their proposed criteria and the aspects discussed within the papers themselves. They have found that work compatibility criteria is connected with the presence of problem concepts, development, and the involvement of stakeholders in the validation process of an approach. Also, they found that usefulness can be identified by the involvement of stakeholders in the validation process of the approach. Ease of use was found to be related to the development aspect. Based on that, observations were made regarding how each optimization approach under their study is related to and apply the attributes of usefulness, work compatibility, and ease of use. Their observations and findings can be found in the next Subsection 2.3.2.

The study presented in [5] covers the research papers published between 1993 and 2013 that can potentially be considered as a Search-Based Software Project Management (SBSPM) approach. Their study aims to identify the categories and the effectiveness of the SSSP optimization approaches as well as to provide directions for future research. In [5], several project management aspects are identified to categorize the approaches that solve SE problems including SSSP. They have linked each SSSP approach under their study to a management aspect based on the text extracted from the formalized problem addressed by each. The findings of their study are discussed in the following Subsection 2.3.2.

2.3.2 Observation and findings

2.3.2.1 Categories of SSSP Optimization Approaches

The main categories of the optimized SSSP approaches can be illustrated from the study presented in [5]. This study has defined two categories of software project management optimization that any optimized SSSP approach can falls within, which are effort estimation, and scheduling and

staffing software projects, depicted in Figure 7. From the figure, it can be seen that 55 papers have discussed in general the optimization of software project management problems. In addition, Figure 7 shows that approaches that optimize scheduling and staffing software projects until 2013 are only about 28, and also effort estimation is in equal interest with the same number of published papers.

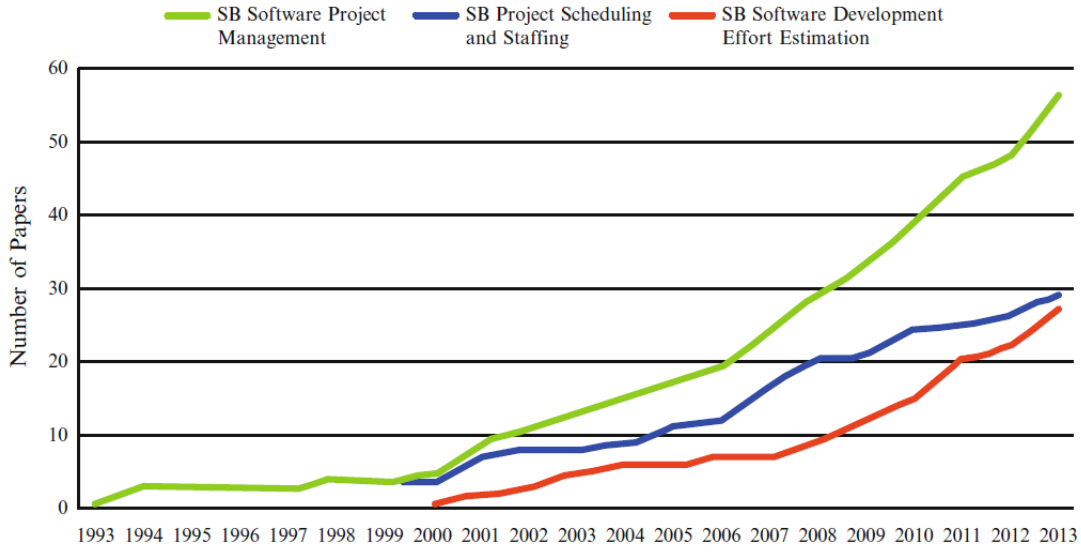

Figure 7: Date and Number of Optimization Approaches illustrated from [5]

However, the approaches used by [5] are categorized into four groups. These categories are minimization of project completion time, risk based approaches, overtime planning approaches, and software development effort estimation approaches. Overall, the study presented in [5] is clearly focused on overviewing the area and provide a taxonomy for the Search-Based Software Project Management SBSPM, and concluding the approaches proposed to solve the software engineering problems within the Search-Based Software Engineering SBSE term.

2.3.2.2 Attributes of the human resource allocation optimization approaches

The study presented in [24] provides a bird's eye view of the software project aspects that the SSSP approaches are taking into account. The main aspects according to [24] are project, artifact, task, resource, team, and skill. These main aspects are illustrated by [24] and presented in Figure 8. From Figure 8, it can be seen that availability, dedication, and salary are the main attributes illustrated within the resource aspect. In addition, the main attributes of project tasks aspect are the estimated effort, precedence relation to another task(s), and duration. From their representation, it is clear how teams are connected to resources, and skills are represented as a

connection between the task and resource aspects. The representation of HRA problem by [24] in Figure 8 shows that the objectives defined by the SSSP approaches are project duration and cost.

Limited optimization approaches address skills and competencies of human resources [5, 24]. Artifacts such as dependencies and variability of size of project tasks are also addressed by limited approaches [24]. Also, limited approaches concentrate their allocation to teams and only 13 approaches perform their allocation to teams [24]. According to [24], there is still a room for improvement to bring the approaches closer to the industrial environment regarding the attributes and factors that are related to technologies, humans, the development process, and organization aspects too.

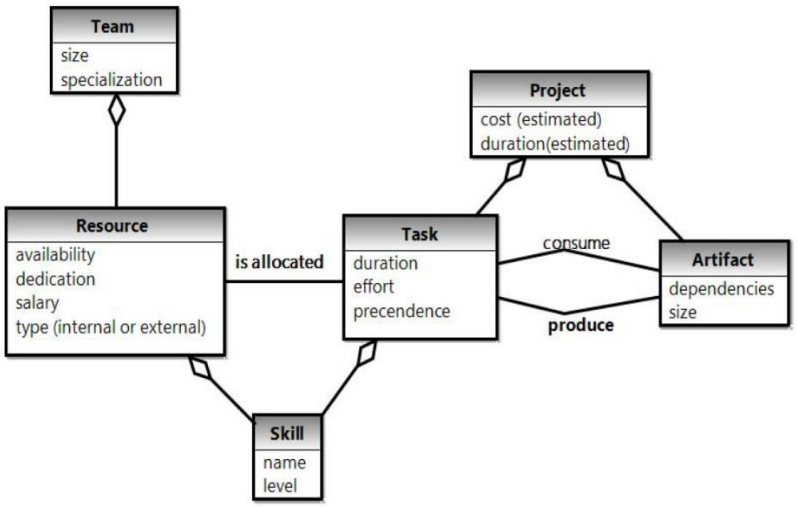

Figure 8: Human resource allocation problem illustrated from [24]

2.3.2.3 Optimization Techniques and Validation Methods

Both studies in [5, 24] have stated that so far the most adopted optimization technique is the Genetic Algorithm (GA) with 80% usage amongst their study subjects. Moreover, only eleven approaches use metaheuristics techniques such as Particle swarm, Ant Colony, and simulated annealing [24]. The objectives in most optimized solutions (approaches) were found to be concentrated on project completion time and project cost [24]. Moreover, approaches studied that belong to minimizing software project completion time found that most combine other objectives such as cost and quality or even multi-objective optimization [5]. In addition, few of these approaches used an empirical study to evaluate their approaches or real software project data [24]. Approaches that have used Quasi-experiment, using the opinions of the targeted population (PMs) typically used small sample size of data, and had a restricted participation of the main stakeholders that do not reflect a real software project environment [24].

2.3.2.4 Nominated Optimization Approaches for industrial adoption

The study presented in [24] claims that the approach presented in [28] is the only approach that has a proper structure to cover the attributes, and overcome the issue related to industry problems. In addition, they claim that the approach presented in [16] is the only one that can overcome the development issue by having the ability to integrate with a management tool. Moreover, they claim that the approach in [16] is the only approach that has been validated by a quasi-experiment in a proper manner. Overall, there are limited number of approaches that have been addressed by both studies, which are [14, 20-22, 28, 94].

2.3.2.5 Possible future trends and research directions

Based on the observations stated in [24], there is a need for more research on the attributes that bring solutions closer to software industry environments as most of the optimized approaches represent the problem concept by a limited number of attributes. In addition, most of the approaches that were subject of [24] study have inadequate development in GUIs, and they fail to address integration with other project management tools. In addition, empirical evidence must be gathered and evaluation has to be done to validate the optimized approaches. Accordingly, the study in [24] observed that a lack of evidences about usefulness of the approaches is exists, and the SSSP problem addressed by the approaches have to address the real industry environment.

The study presented in [5] concludes that far more research is required to address the allocation of resources in software projects. According to them, promising areas using optimization techniques to solve the problem of staffing and scheduling software projects include:

1) Interactive optimization: This kind of optimization explores computationally the expertise of project managers by which how they perform different management objectives.

2) Dynamic Adaptive Optimization: this kind of optimization helps the decision makers to interact continually and dynamically in real time to explore the implications of their decision.

3) Multi-Objective Optimization: this kind of optimization focuses on decision support in complex multi-objective problem spaces to include and combine several objectives together.

4) Co-Evolution: this kind of research is concerned with modelling the fitness between two populations of the solutions obtained by the optimization. Here, they suggest that increasing in one population should affect the others by reducing their fitness. This kind of optimization accordingly could lead to better and faster solutions, which requires more attention by researchers.

5) Software Project Benchmarking: the most challenging problem in the human resource allocation optimization is the lack of real world project data. Accordingly, having a real dataset to benchmark the optimization approaches has emerged as an important aspect to this field.

6) Confident Estimates: estimation in management processes is considered as the most difficult aspect. Noisy and uncertain inputs are the basis of this problem. Accordingly, introducing levels of uncertainty within the estimation process and measure their effects can be a promising area for further investigation.

7) Decision Support Tools: so far, the optimization approaches are all under research. A promising area and trend in human resource allocation optimization is to develop these approaches as decision support tools and make them available to transfer the most successful methods into practice.

2.4 SSSP Optimization Approaches

The solution process of the SSSP problem requires consideration of a range of factors, such as task precedence relationship, skills, and effort [24]. These factors are considered inputs to SSSP approaches, whereas the output(s) of these approaches typically consists of minimization of project time and/or cost, or maximization of resource usage and skill availability in projects. These factors are used by SSSP approaches to mathematically formalize the problem, where the formalization is then used to model the solution using an optimization technique. As the problem of SSSP is an NP-Hard problem, Meta-Heuristic techniques can be the accepted solution by approximating the results. With diversity of Meta- Heuristic techniques that can be used to solve the SSSP problem, comparison between their effectiveness to software engineering problems is important to be addressed [10]. Yet, studies that evaluate the techniques for SBSPM problems are limited as in [22]. In [22], a comparison between the adopted technique and others is performed and concluded that SA may outperform GA in cases of absence of dependencies between the tasks, whereas in cases of having dependencies, the study shows that both GA and SA perform the same.

The Meta-Heuristic techniques that have been used to solve SSSP problem, by many approaches as in [22, 28], are only GA, SA, and PSO, and so far, 80% of the optimization techniques used by the optimization approaches are Genetic Algorithm (GA's) depicted in [95]. Some of these approaches however, combine different techniques together to obtain good results. This can be seen in the approaches in [16, 28, 96]. However, using a heuristic technique for GA population initialization can narrow the search space as the search might be trapped in a local optimum[97].

According to the discussion presented in [17], this section will discuss and detail the SSSP approaches' aspects as follow. The first aspect is the formalization of each approach with consideration of inputs, and constraints. The second aspect is the solution proposed considering the representation of problem, initial search population, the technique and its stochastic process adjustment, fitness function, candidate selection, optimizer's settings, and its validation results.

2.4.1 Problem Input Formalization

Project Decomposition and Effort Estimation

Activities within software development projects can differ from company to company and depend on whether sequential, incremental, or iterative development is used. Project tasks can be organized according to which iteration or phase the task belongs to. In addition, project tasks can have dependencies, for example, when a database needs to be completed before data retrieval can be tested. Most modern software development methods deploy a phased approach and it is therefore no surprise that most SSSP approaches seek to optimize resource allocation within such as context. Approaches such as [16, 23, 28, 91, 96] assign the team members to a particular activity of a development phase, such as requirement elicitation, analysis, design, implementation, or testing. However, between these approaches differences can be observed in the phases supported. For example, [96] only considers design, implementation, and testing activities. The approaches in [28] and [23] consider the activities of analysis, design, implementation, and testing. The approach in [91] is more focussed as it only support two activities, which are implementation, and testing. Optimization approaches that explicitly consider the iterative character of software development methods are more rare with [28] as one of the few published results. Overall, these approaches have the unit of allocation in common. This unit is the phases that belongs to the increments and iterations of the software project.

On the other hand, there are considerable number of approaches such as [15, 18, 20, 22], which do not perform the resource allocation optimization for a particular methodology. This is due to the diversity of methods in practice and to generality of use that these approaches are seeking. The one in [18] is considered by [5] as one of the first approaches proposed in minimizing software project completion time. This approach deals with a situation where a software project has to be developed having interdependent tasks each of which requires a skill set that should be possessed by the assigned resource(s). It is obvious here that the problem the approaches are representing is also important to identify. The problem represented in [18] has also been addressed by a wider

range of approaches as in [14, 15, 22]. Approaches that perform the allocation as these ones do, but without consideration of skills can be found in [20, 21, 94]. These approaches focus on assigning the resources to tasks without the description of the content, and regardless of the required skills and unit of allocation. Noteworthy that these approaches consider the unit of allocation as a project task regardless the description to which phase or activity nature that this task belongs to.

For both units of allocation, effort estimation is required. While a range of estimation techniques and tools are available nowadays, these techniques have shown their maximum capability with less accuracy and therefore each approach has just presented the estimation of effort by introducing the COCOMO. Broadly speaking, COCOMO [98] has the nomination amongst the estimation tools, and partially these approaches have proposed the use of it. Others purely mention that the effort of each unit has to be estimated in term of Man-Month as in [28], or Man-Day as in [14]. For more information on how this value can be estimated see Section 2.1.1.

Task Dependencies

Dependency between project tasks is the subject that is considered in different ways by the optimization approaches. The approaches in [20, 21] for example ignore the dependency between project tasks. The approaches in [14, 15, 18, 22, 94] on the other hand consider dependencies as one of the problem inputs. This dependency can be illustrated as precedence relationships existing between the project tasks, and a task cannot be performed before its predecessor. Representation of dependency in general has been used in these approaches by the Task Precedence Graph (TPG) for a direct acyclic representation -see Section 2.1.2- . Others as in [16, 28, 91, 96] consider the phases, increments, and releases of development as a natural dependency identifiers that have to be defined within the problem inputs. Approaches that combines both descriptions can be found in [23]. This approach in addition to the phases of development, has the ability to deal with the situation where dependencies between project tasks are represented in the form of TPG.

Single and multiple projects

Staffing and scheduling software project optimisation approaches can be divided into two categories based on the scope of projects that they consider. The first category of approaches addresses the optimisation only for a single project at a time. The second category considers the allocation for an entire organizational environment consisting of multiple projects that need to be performed at the same time.

In the literature, the overwhelming majority of proposed approaches target the optimisation of human resources considering only a single project problem consisting of multiple tasks as can be seen in [14-16, 18, 20-23, 28, 91, 94, 96]. Each one of these approaches considers the project task as a standard unit for allocation, and performs the allocation to project tasks either for teams or for individuals. The optimization of human resources for multiple dependent development projects has received considerably less attention in the literature, with the work by [99] as an example of a multi-project approach. However, even this approach simplifies the problem by modelling the multiple projects as multiple dependent tasks. In addition, this approach ignores the longer-term considerations such as project overlap. It can be concluded that the category of multi-project resource allocation optimisation requires further attention beyond the current state of the art.

Human Resource Properties

The assignment of human resources depends upon three factors. These factors are skills and competencies, productivity, and availability of human resources at the time that the project will be performed.

Competencies of human resources in software projects play a key role in effort planning and resource selection. A selection that is based on competencies searching for the best resource who can perform a task with high-quality, and shortest time, can also be considered in team formation. For this reason, a number of competency models are proposed to support the manager in the selection process. Competency models such as [57, 62] can be used to form teams according to specific and important attributes of resources that lead to identify the best resource to perform a particular team role. Another model presented in [7] named as Best-Fitted resource model uses a relationship ability matrix. This matrix defines a relation between the skills of available resources and the ones required for a project. This model helps to identify what skills are similar to each other, which can lead on understanding of the capability of a resource, having these skills, to learn new ones.

Approaches that consider human resource skills through their optimization process as in [14, 15, 18, 22, 91] did not clarify the competencies used in their selection process except the approaches proposed in [16, 23, 28, 96]. On the other hand, some approaches ignore the importance of human resource skills as in [20, 21, 94] assuming that all the available resources have the same capabilities and competencies.

Productivity on the other hand is represented by the optimization approaches according to different criteria. Productivity can be ignored, estimated based on experience, or based on the

software development activities [16]. Optimization approaches that ignore productivity as in [14, 15, 18, 20-22, 94] assume that the resources share the same expertise and productivity. For this reason, they just count the number of resources as a measure of team productivity. Approaches that consider productivity of resources based on their expertise and skills as in [16, 23, 28] model the resources based on specific programming languages, and activity experience. Productivity of a resource in these approaches can be twice or thrice than of a normal resource. In addition, some approaches still use the standard productivity metric for development, and debugging productivity as the one presented in [91], which depends upon how many line of code the resource can develop within a unit of time.

It is clear that productivity is a problem with no agreement as to how it can be measured. Some of the approaches in SPM consider the role that the resource could play within a team. They measure productivity according to these roles such as analyst, designer, developer, or tester. However, the resources in software projects can play different roles and they possess more than one professionalism. In addition, the resources in software firms can be specialized in a particular software technical development and/or product according to the department they work for.

Availability on the other hand is the subject that most of the optimization approaches consider as a predefined input. This input is identified by the approaches in three different manners. The first one is assuming that a number of resources are available as in [20-23, 91, 94, 96]. The second manner is by using a percentage to express their availability as in [14, 15, 18, 28]. The third manner is by using a time window to express the availability period of a resource. This manner is addressed by a few approaches as in [16, 93] using a period that expresses unavailability of a resource during the project time. For a single project optimization, the first two manners are realistic. However, if the problem is at organizational level that consists of multiple projects at a time, then it is important to identify which resources are available, and which are not, for each project.

Team Formation

Many human resource models have paid attention to team formation in software projects as in [57, 62]. These models focus on skills and competencies of the human resources in software project for each team role such as analyst, designer, programmer etc. A team that requires different roles combining different skills, competencies, and disciplines of software development all to be in the same team including stakeholder, is called a cross functional team [100]. Since the late of the 20th century, this kind of team has been practiced in industry, and addressed by software development methods such as Agile [100] and DevOps [101]. However, this kind of team

formation has not been addressed by any optimization approaches so far. Consequently, this kind of team formation should emerge and be addressed by future approaches.

The team formation considered by the optimization approaches is the one that assumes all potential members share similarity in their competencies and skills. This kind of team formation is adopted by the approaches in [22, 23, 28]. Noteworthy that approaches as in [20, 21, 94] do not consider skills and competencies and they form the teams randomly. Moreover, some approaches as in [14-16, 18, 91, 96] do not consider team formation but rather to perform individual allocation to project tasks.

2.4.2 Constraints and Penalties

Constraints on human resource allocation in a software project are the most important part in the allocation. This is due to the reality that dependency between tasks, the number of the resources, their skills, and availability can affect the development time, quality, and cost of a software project. Optimization approaches that consider this reality are in [14-16, 18, 22, 23, 28, 91, 94, 96] have the concern about these constraints for how they can be integrated in their approaches and how it will affect the outcomes. While these approaches are combined with a meta-heuristic optimization technique, the outcome solution is subject to change stochastically. That implies the constraints in these approaches can be violated by the optimization techniques while searching for local or global optima. On this occasion, the approaches offer penalties, which are used to revise the outcomes by adding a penalty value to the final result. Optimization approaches that consider penalties can be seen in [14, 15, 18, 28, 91].

Optimization approaches that consider a constraint on precedence relationship and dependencies between project tasks are in [14, 15, 18, 22, 23, 28, 91, 94, 96]. Project task dependency is represented by [28, 91, 96] as a precedence relationship between the features or modules of the software. The constraint is that no feature or module can be offered by a release that contains the one preceding it as well as tasks of a feature or module should be organized based on the phase's sequence. The approach in [22] represents the dependency constraint as that each task depends on another, and this should be implemented as a queue of tasks, that no task should be offered before its phase. The approaches presented in [14, 15, 18, 22, 23, 94] on the other hand have combined the task precedence dependency constraint within their solution. These approaches employ for reliable outcomes a simulation of schedule that in each step will verify whether the precedence relation is met by the current solution or not.

Optimization approaches that consider a constraint on the number of resources allocated to a single task are in [18, 22, 28, 91, 96]. The approach in [22] has the constraint on the number of teams allocated to a single task. This constraint is that for a single task only one team should be allocated to it. Similarly, the approaches in [91, 96] made the constraint that for a single task one resource should be allocated to it. On the other hand, the approach in [28] made a restriction on the number of resources allocated to a team. The number of resources should not exceed the value that the manager defines for teams according to the project environment.

Optimization approaches that consider a constraint on the resources skills are in [14-16, 18, 22, 23, 28, 91, 96]. The approach in [28] even classified the resources according to their skills as expert or novice. In addition, this approach impose a restriction that each team should have at least on expert. This constraint is to maintain the quality of development, since using novice workers could affect the work quality.

Optimization approaches that consider a constraint on sharing the resources are in [22, 28]. In these approaches, sharing resources between multiple teams is not allowed. The approach in [28] however, made an exception for the testing activities in software project to be done by cooperating part or all the resources together. In addition, the approach in [22] made additional constraints; that each team should share the same expertise and at least one team should cover the expertise required for the project. Moreover, the approach in [28] considers an additional constraint on the incremental level, this constraint is to ensure that the resources are continually involved in the module they are working on within the increments, so there is no need for extra time for understanding the module.

Penalties on the other hand are considered by the approaches in [14, 15, 28, 91]. The approach in [28] penalizes the violation of the following constraints adding a value that should be defined by the project manager. These penalties are made on the incremental, novice teams, dependencies of the phases, and the number of developers within the team constraints. On the other hand, penalties on violating the constraints in the [91] approach are combined within the project cost. These penalties are on both development and debugging productivity, as well as the error rate. The approaches in [14, 15], penalize only for skills as the scheduling is combined within the fitness function. Skills penalty in [14, 15] is implemented by counting the number of missing skills, and multiply this value by the overall effort of the whole project, to penalize the project time to end with a very high value.

2.4.3 Solution Representation

Modelling the inputs of an optimization problem is the first step of the optimization process. The second step then is to identify the goal for the optimization, which is represented as an objective function. In addition, in some cases the optimization of an objective should be performed with respect to constraints that are required to be satisfied. For some optimization problems constraints can be violated while the technique performs a probabilistic process to obtain solutions. Therefore, the optimization process applies penalties on the fitness value in cases of violation.

The process adopted by the optimization approaches to solve SSSP can be seen as a multi stage process. These stages are considered either to ensure the completeness of the solution proposed or to reduce the processing time of the optimization process. Optimization approaches as in [14-16, 18, 20-23, 28, 94] search by a single or multi stage the possible optimal or near optimal results based on the adopted allocation process. Stages in these optimization approaches can be considered as a case of team or individual allocation. Some of these approaches propose the allocation of individuals directly to tasks as in [14-16, 18, 23, 91]. This means that a single stage is required to allocate those resources to project tasks. On the other hand, others as in [20-22, 94] form the team and sort the tasks first and subsequently assign those teams to project tasks. Only the approach in [28] uses two stages to perform first team level allocation that assigns each team to a group of tasks, and the second stage is to assign each team member individually to tasks from the allocated task group. However, while their final solution is encoded into a 5-D allocation matrix without consideration for sharing resources across teams or tasks, that means the allocation performed in this approach at the end is an individual allocation.

The solution structure of an approach can be represented in different ways according to the technique used. For genetic algorithms, the representation is by a “chromosome”. This chromosome contains a finite number of “genes” that each represents an element of the solution. In the problem of SSSP, an element can be a resource, a task, or a team. The mainstream of the approaches employs GA in their solution for fast and accurate outcomes. Chromosomes can be represented as a vector as in [20-22, 94]. Others as in [14, 15, 18] use 2-Dimensional array to represent their resource allocation solution combining the resources and tasks. Similarly, approaches as in [23] use a cell array to accelerate the computation time of the algorithm. Chromosomes moreover can be encoded using different systems such as binary, permutations,

value, or tree. The reader can refer to [74, 84-86] for information on chromosomes, their structures, and encodings in GA.

The approaches that use SA on the other hand can have the solution structure similar to those for GA representations. For instance, the approach presented in [28] combine five allocation attributes in their problem into a multi-dimensional array. These attributes are resources, modules, increments, phases, and time slots of allocation elements represented by a 5-D matrix. Others as in [21] used the same structure for hill climbing, SA, and GA optimization techniques.

Initial Population

Initial population is the starting point of the search in meta-heuristic techniques. Two types of initial population creation can be used, which are random and heuristic initialization of population using heuristic optimization techniques to define the starting population [86]. Broadly speaking, approaches that use GA attempt to initialize the population randomly, as the use of heuristic techniques might lead the solution to a local optimum. These approaches can be found in [14, 15, 18, 20-23, 94]. However, there are approaches that combine a heuristic technique such as Greedy to initialize the population as in [28]. This attempt was claimed to have improved the search time, as well as the solution quality.

Stochastic Optimization Process

A stochastic process in Meta-Heuristic techniques provides a mechanism to create new solutions by adjusting heuristic changes within the old ones. Stochastic processes in GA are combined with two operations, which are crossover and mutation. Crossover operation perform modification on the chromosomes by exchanging subparts of two chromosomes and combine them into a child one [86]. Different operators for crossover can be used according to which subparts of parents' chromosomes should be selected for the new child. The most popular crossover operators are single-point, two-point, uniform, and arithmetic [86] [102]. The reader can refer to [86] [102] for more information.

The mainstream SSSP approaches that employ GA within their solution modify their crossover operators according to their solution encoding and structure. The approach in [94] uses the basic single-point crossover. The approaches in [20-22] however, modified the single-point crossover for the subparts selection and order of the exchanged chromosome's elements. As the approaches in [14, 18] use a 2-D chromosome structure, their crossover operator performs a single-point crossover but with consideration of rows and columns to exchange. The one in [15] on the other hand performs a modified crossover that works with equal probability whether to exchange the

rows or columns of their 2-D structure. The one in [23] moreover, performs a uniform crossover on both parents' chromosomes to generate a child one.

For the approaches that use SA, this technique provides perturbation operators to manipulate the subparts of the solution. Perturb operator can be by exchanging two elements of the solution, or by moving a single element value from one to another. Both perturbations are used in [28] with equal opportunity. These operators allow the SA to advance the search into the global area than its basic hill climbing technique.

To avoid same solution generation and explore a global area in the solution space, mutation can be used in GA to randomly perform changes on the solutions [86, 103]. Mutation in GA can be done either by selecting and exchanging values randomly to some solution elements, bits flipping for binary encoded chromosome, scramble or inversion of part of the chromosome, or swapping values between different elements of a chromosome [102].

In [14, 18] approaches, a single element bit-flip is used. The approaches in [20-22] provide solution with two representations, one for team assignment, and the other one for task allocation. Mutation in these approaches works by exchanging two elements of the chromosome for the task allocation representation, and a single element exchange for team assignment representation. The one in [15] uses a different mutation strategy. Their mutation works by assigning probability to each element to randomly assign a new value from the range they have defined as a dedication of the resource represented by that element to the allocated task. The one in [94] moreover, uses two mutations as their approach combine two solutions for team assignment and task allocation. The team representation combines random generation mutations for all chromosome elements. Mutation for task allocation representation on the other hand works by randomly setting a value for a random chromosome's element. Similarly, the one in [23] approach mutates the solution by exchanging a chromosome element with a random value.

Selection of Candidate Chromosomes Solutions

Selection of candidate solutions as one of the heuristic operations selects the fittest chromosomes among all the permutation produced by the algorithm. These selected solutions will be used to produce a new population using different methods [86]. These methods are roulette wheel, stochastic universal sampling, tournament, steady state, rank, elitism, and random selections [86].

Both [18, 20] use elitism selection. Approaches in [21, 22] use roulette wheel selection mechanism, and tournament for NSGA-II of multi-objective for [22]. In [14, 15, 23, 94] approaches, tournament selection is used.

Objective Function

Depending on the problem that the approaches solve, the optimization can be either for a single or multi objective(s) function. Single objective approaches as in [20, 21, 23, 28, 94] search for an optimal, or near optimal project completion time solution. Approaches in [14-16, 18, 22, 91, 96], on the other hand optimize the allocation for multi-objective functions. These functions simulate the Decision Maker's (DM) choices for producing high-quality allocation that satisfies part or all project stakeholder objectives. For example, the approach in [16] optimizes the allocation as follows: When the project only requires professionals then the most qualified team is the objective function. In case of shortage of resources, and the available team having minimal skills, then the minimum qualifying team is the objective function. In addition, for project time the DM can select the fastest team. For cost purpose, DM may select the cheapest or smallest team. DM on the other hand can consider a best partial solution when the resources do not satisfy the requirements for the activities. This can be seen as an example where an approach optimizes the allocation and can separate the objectives from one another. The one in [18] moreover optimizes for four objectives. However, two of these objectives, which are overtime work of resources and job assignment validity are used within their scoring model to penalize the infeasibility of a solution adding high value to the fitness functions of project time, and cost.

However, there are approaches that do not perform the optimization considering separation between the objectives as in [91]. Objectives in this approach are combined together so that the whole process of optimization can be seen as a single objective solution.

Approaches that optimize the resource allocation for the objective of minimizing the completion time of a software project are in [14-16, 18, 20-23, 28, 91, 93, 94, 96]. However, for part of these approaches project time estimation is differing from one to another. This is depending on the input model they use. For instance, some models use the term of time windows, which means the time of each task has already been defined, or predefined with start and end task times. Therefore, these equations cannot be seen equally with other models that use effort to estimate the overall project time span.

Optimizer Settings

Each SSSP approach considers different values of GA parameters for the chromosome, population size, generation, mutation probability, etc. The value of these parameters however play a key role on the outcome quality and time spent on searching for an optimal or near optimal solution. For example, if the generation is set to a high value, that means the GA will constantly keep producing populations and searching for the optimal one, even if the very best solution has been found, the search will continue till the number of generations is reached.

Population size by many approaches is set to be 100 as in [20, 21, 23]. Both [14, 15] however, set their population to 64. The one in [18] sets to 60 population, and both [22, 94] set their population to 50.

The number of generation for a GA on the other hand, which constitute the process of moving from one population to another acting as a termination criteria [97], is adjusted differently in each approach. Approaches that search with high number of generations might have some implication on the runtime as in [14, 18, 20, 21, 23]. The approach in [14] sets the generation to 5000, the approach in [20] sets to 1000, where [18] set to 500, and both [21, 23] set their GA generation parameter to 400. Others have set their generation parameter considerably less than the previous approaches as in [22] approach to be 250, the one in [94] with 100, and [15] to 79 generations.

Moreover, the mutation operation, which involves creating the next population, is usually works with low probability [97]. The approaches in [23] mutate with very low probability of 0.05. The approaches in [20-22] set their mutation probability to 0.1. The approach in [18] mutate with probability of 0.15, and the one in [94] set the mutation probability to 0.2. However, there are some approaches that modify their mutation rate according to the problem size as in [14, 15]. Their mutation rate is calculated according to the number of tasks and resources within the solution.

Probability of crossover, as for creating new chromosomes based on both parents, varies in the approaches that use GA. The approaches in [20, 21, 94] set their crossover probability to 0.6. The approach in [18] sets the crossover probability to 0.65. The one in [22] set to 0.7. The one in [15] set to 0.75. Finally the one in [14] sets the crossover probability to 0.9, and the one in [23] set their crossover probability to 1.0.

The most critical operation that could affect a GA-Based optimization approach is mutation. In some cases it could lead to a better solution, however, mutating the solutions can also leads to explore far more solutions than the best one, which will accordingly have implication on the computational time and the results' stability of the approach. For best GA results, the approach in

[18] concludes that the population size should be in range of 50 to 80, and crossover rate between 40 to 80 percent, and mutation rate between 10-40 percent.

The SA technique is adopted by the approaches in [21, 22, 28]. However, the approach in [22] used this technique to compare its results with GA, and HC. The configuration of this technique's parameters in the approach presented in [28] for the initial temperature, number of internal loops, number of external loops, parameter control, and cooling factor were are 100, 500, 8, 2000, and 0.95 respectively. On the other hand, both papers of [21, 22] do not state the values of these parameters that should be an important subject to their evaluation of the techniques adopted.

2.4.4 Validation

Validation is the key for presenting the quality of an optimization approach. Different methods are used to validate the optimization approaches. Some of the optimization approaches adopt experimental methods to compare their approach with different optimization techniques. Approaches that compared their results with other optimization techniques are in [21, 22, 28, 96]. The approach in [28] for instance made an empirical analysis between SA and Greedy optimization techniques using the same data. The performance on average for the obtained results was that the approach using SA outperform Greedy algorithm. The one in [15] in addition, performed their evaluation on different Evolutionary Algorithms (EA) and approaches including the one in [14]. Their findings suggest that their solution proposal outperforms the others especially using an improved EA called Pop-EA.

On the other hand, some of the approaches as in [20, 91] performed their validation using sensitivity analysis, which investigates the sensitivity of the outcomes by changing the input and the attribute values. For example, the approach in [91] claim that:

1. Productivity does not necessarily lead to reduce development time as expected.
2. Higher productivity does not guarantee an ideal software development outcome.
3. Increasing the demand only on high quality or for both productivity and quality to be moderate can lead to better results than to consider both factor to be high for ideal outcomes of the software development projects.

Moreover, few of the approaches as in [16, 23] performed an empirical study validating their approach by observing how participants can do the allocation, and the resulted quality of their allocation. The approach in [16] performed this kind of study to check their understanding of the

field. However, their study has 16 participants who were all students, and their conclusion was that 22% of the participants were able to provide a solution close to the one of their model.

The validation also requires a dataset to test the applicability of the optimization approaches. Approaches that used real software project data as in [21, 22, 28] used the data to show the effectiveness of their approaches. On the other hand, other approaches used hypothetical "simulated" data in their validation as in [14, 18] [15, 16, 23, 91, 96].

2.4.5 Selected SSSP Approaches for Benchmarking and Comparison

As benchmarking research should not bring as many approaches as available to one comparison, a representative set has to be identified [104]. This set however, should include approaches that partially or fully close to and suitable to software project time and SSSP problem defined in Section 1.2. These approaches moreover, should have wide generality of usage. For instance, approaches that focus on solving project time minimization for a particular development method should not be included. A limited number of approaches that consider this aspect, however, can be found amongst the approaches proposed for SSSP problem. The selection criteria for the SSSP approaches adopted for the work carried out for this thesis are as follow.

1. The first criterion is the approach should at least be among the most cited and referenced ones. This task was completed by referring to both studies in [5, 24] where their findings and results contributed to the selection for this chapter.
2. The focus is on the approaches that perform a single objective of project time span minimization. Noteworthy that some of the approaches use multi-objectives optimization. If the approach combines the time into a weighting scoring model, then it is possible to give all the scores only to project time so it can be easily calibrated to project time objective. In this case these approaches can be included in our benchmarking and comparison study.
3. The approach moreover, should use the effort estimation for project tasks, and not any consideration to time window, or task time frame consisting of start and end time. Estimated effort can be obtained using different tools and techniques, but the end result should be in terms of man-month, or man-day. However, both terms are similar and for accuracy purpose Man-Day will be used.

4. Mainly to make the approaches comparable they have to adopt soft constraints. In addition, dependencies between project tasks should be formed in term of which one can start before the other, which can be represented by a directed acyclic graph –see Section 2.1.2-, or any dependency sorting mechanism with TPG. No matter which software development method is used, projects always have dependency either directly between the tasks, between the phases, or can be between iterations or increments that should be plotted by the approach.

The approaches that comply with this criteria are more likely to be included within the representative SSSP approaches set. Accordingly, the SSSP approaches selected for the evaluation and comparison are mainly chosen for the following reasons: relativeness to SSSP problem, publication closeness to software engineering, and possibility to adjust for single fitness outcome.

The first approach to be selected is the one presented in [18]. This approach is considered as one of the earliest work on optimizing software project time. This approach, in addition, complies with the earlier mentioned criteria, and has a scoring model to measure the fitness of project time and cost. Another approach that is selected for its unique proposal and the adaptation of our selection criteria is the one presented in [20]. This work focuses on the team distribution and is the base for the works presented by [21, 22, 94]. So, including this approach could provide a clear evidence of the usefulness of the adopted processes and procedures in terms of allocating the resources and estimating project time. As this work is a base for other approaches, these approaches are also important to be included to test their improvements and to what extent they could deliver an optimal or near optimal solutions in terms of project time minimization. Accordingly, the approaches in [21, 22, 94] are selected to our benchmarking and comparison study. Searching for an up-to-date approaches has, in addition, come with several approaches. However, many approaches have similarity in their proposals, and for many cases limited description is found that allow for reproduction. The approach in [15], on the other hand, is found to have some improvement over the approaches in [14, 18]. This approach, in addition, complies with our criteria, which makes it one of our choices for the benchmarking and comparison study. Including this approach could offer a quality for our comparison study against these that it based on and outperform. Moreover, [28] approach has been mentioned in many proposals such as [5, 23, 24] for its complexity on offering too many constraints that could affect it solution quality. To put these claims into test, this approach is accordingly selected to our study. It is noteworthy that this approach complies with our criteria. The last approach to be selected while searching for a new

up-to-date is the one in [23]. This approach complies with our criteria and offers a different resource allocation pattern and GA representation. This approach is, to the best of our knowledge, the first to use cell array for GA representation, which makes it a choice for the comparison against the other proposals. While many others could possibly be a quality for our study to test, many of these approach that have not been selected share similarity in almost every detail of the resource allocation and optimization process, or fall in the gap of "out of criteria". This can be where different effort estimation units, or time measures are used, such as the time window frame, having the assumption of task time as an input that has already been measured, and what left for the optimizer is to align what every possible tasks together.

Table 9: Selected SSSP approaches

Approach	Published in
Chang01	[18]
Antoniol01	[20]
Antoniol02	[21]
Alba01	[14]
Ren01	[94]
Kang01	[28]
DiPenta01	[22]
Minku01	[15]
Park01	[23]

The following Table 10, in addition, highlights the main aspect that these approaches consider. Table 10 depicts the approaches considered for the benchmarking and evaluation with respect to the aspects of problem inputs including project tasks, their precedence relationship, skills, and productivity of resources. In addition, it can be seen in the table the type of constraints, the optimizer that the approaches employ, the objective function(s), and the representation of solution considered by each approach. Moreover, the table illustrates how the approaches have been validated in terms of methods and data used for this purpose. Methods of validation are illustrated in the table by three letters. These letters and what they stand for are (C) for comparative analysis, (S) for sensitivity analysis, and (E) for Empirical evaluation. In addition, the type of data used for the validation illustrated in the table by (S) for synthetic, and (R) for real-world data. For instance, this table shows that the approach Ren01 considers two inputs to the problem which are task and precedence relationship, and incorporates them in the GA optimizer by 2 vectors in the solution representation of project time minimization objective. From Table 10,

it can also be seen that the validation of this approach is carried out by sensitivity analysis (S) method using a real-world dataset (R).

Table 10: Attributes of Selected Benchmark SSSP Approaches

SSSP Approaches	Problem Inputs				Constraints		Optimizer	Objective(s) of minimizing Project	Solution Representation	Validation	
	Project Tasks	Precedence relationship	Skills	Productivity	Hard	Soft				Method	Data
Chang01	✓	✓	✓			✓	GA	Time, Cost	2-D	S	S
Antoniol01	✓						GA	Time	2 Vectors	S	R
Antoniol02	✓						GA	Time	1 vector	C	R
Alba01	✓	✓	✓			✓	GA	Time, Cost	2-D	S	S
Ren01	✓	✓				✓	GA	Time	2 Vectors	S	R
Kang01	✓		✓	✓		✓	SA	Time	5-D	C	R
DiPenta01	✓	✓	✓			✓	GA	Time	2 Vectors	C	R
Minku01	✓	✓	✓				GA	Time, Cost	2-D	C	S
Park01	✓	✓	✓	✓			GA	Time	Cell Matrix	E	S

2.4.6 Detailed Description of the Selected SSSP Approaches

The algorithm named Chang01 starts by initializing population P of size 60, where the solution is represented by a 2-D matrix chromosome structure. The rows in this chromosome represents the resources and the columns represents the tasks. The value of each cell of the matrix represents a percentage that the resource will participate with between {0, 0.25, 0.5, 0.75, 1}. The GA repeats by 500 generations three main processes, until a solution satisfy the objective function. The first process is selecting parents from P using Elitism selection. For the selected parents, the second process performs a single-point crossover with probability 0.65, and mutation using a single element bit-flip with probability 0.15, then stores these new solutions. The third process is evaluating the solutions based on the fitness function having the best to survive for the next generation. The fitness function of this algorithm works by calculating the overall participation percentages of resources to each task. The task that is under computation, however, should not be waiting another task to finish. The task that has a precede one, should wait till it finish. The time for this task is estimated by dividing the estimated task effort over the overall participation of all the resources to this task. In order to the participation percentage of a resource to be counted, his/her skillset should meet the ones required for that task, otherwise the resource should not be counted for that task. The fitness function in addition checks whether each resource is working on more than one task at a time, if so, then the solution then the overall dedication should less or equal to one or penalized then with maximum estimated time to eliminate the solution. The tasks are then categorised into groups where each group has those that can be performed concurrently. The longest task among those in parallel is the one that will be added up to the total estimated project time.

The algorithm named Antoniol01 works as a tandem approach. This algorithm starts by initializing population P of size 100, where the solution is represented by two vectors of chromosome structure. The first chromosome is designed for the ordering of tasks in the queue. Each gene in this chromosome represents a task and the allele value represents the position of the task in the queue order. The second chromosome is designed for the team formation of resources. Each gene in this chromosome represents a resource and the allele value represents the team number that this resource is assigned to. The GA repeats by 1000 generations three main processes, until a solution satisfy the objective function for both representations. The first process is selecting parents from P using roulette wheel selection. For the selected parents, the second process performs a single-point crossover with probability 0.6, and mutation using two elements

exchange with probability 0.1, then stores these new solutions. The third process is evaluating the solutions based on the fitness function having the best to survive fort the next generation. The fitness function works for the first representation of task ordering with a uniform distribution of resources to teams. It counts the number of resources of the team responsible to perform each task. Each task time is estimated based on the number of resources in the team assigned to it. Each best solution of the first representation is then fed to the other to search for the best team distribution. What is noteworthy that each team adds up each task time they work on, so the overall estimated project time is then the longest team time.

The algorithm named Antoniol02 starts by initializing population P of size 100, where the solution is represented by a vector chromosome structure. The chromosome represents the ordering of tasks in the queue. Each gene in this chromosome represents a task and the allele value represents the position of the task in the queue order. The team formation of resources is performed at the start of this algorithm randomly, where the number of teams is selected randomly based on the number of resources. The GA repeats by 400 generations three main processes, until a solution satisfy the objective function. The first process is selecting parents from P using roulette wheel selection. For the selected parents, the second process performs a single-point crossover with probability 0.6, and mutation using two elements exchange with probability 0.1, then stores these new solutions. The third process is evaluating the solutions based on the fitness function having the best to survive fort the next generation. The fitness function counts the number of resource for the team responsible to perform the task. The task time is then estimated based on the number of resources in the team assigned to it. What is noteworthy that each team adds up each task time they work on, so the overall estimated project time is then the longest team time.

The algorithm named Alba01 starts by initializing population P of size 64, where the solution is represented by a 2-D matrix chromosome structure. The rows in this chromosome represents the resources and the columns represents the tasks. The value of each cell of the matrix represents a percentage that the resource will participate with between {0, 1/7, ..., 7/7}. The GA repeats by 5000 generations three main processes, until a solution satisfy the objective function. The first process is selecting parents from P using Tournament selection. For the selected parents, the second process performs a single-point crossover with probability 0.9, and mutation using a single element bit-flip with probability 0.01, then stores these new solutions. The third process is evaluating the solutions based on the fitness function having the best to survive fort the next generation. The fitness function of this algorithm works by calculating the overall participation percentages of resources to each task. The task that is under computation, however, should not be waiting another task to finish. The task that has a precede one, should wait till it finish. The time

for this task is estimated by dividing the estimated task effort over the overall participation of all the resources to this task. In order to the participation percentage of a resource to be counted, his/her skillset should meet the ones required for each task the resource is assigned to, otherwise the resource should not be counted for that task. The fitness function in addition checks whether each resource is working on more than one task at a time, if so, then the solution then the overall dedication should less or equal to one or penalized then with maximum estimated time to eliminate the solution. The tasks are then categorised into groups where each group has those that can be performed concurrently. The longest task among those in parallel is the one that will be added up to the total estimated project time.

The algorithm named Ren01 works similar to the tandem algorithm of Antoniol01, however, it combines a cooperative co-evolution algorithm solution by dividing the problem into sub-problems. This is implemented by this approach as two vectors of chromosome structure. The first is concerned with the ordering of tasks in the queue. This structure however, is implemented by the fitness function according to the dependency constraint between project tasks. The value in this vector represents the position of the task in the queue order. The second is a GA chromosome designed for the team formation of resources. Each gene in this chromosome represents a resource and the allele value represents the team number that this resource is assigned to. This algorithm starts by initializing population P of size 50. The GA repeats by 100 generations three main processes, until a solution satisfy the objective function. The first process is selecting parents from P using Tournament selection. For the selected parents, the second process performs a single-point crossover with probability 0.6, and mutation using random generation of new values to all the elements with probability 0.2, then stores these new solutions. The third process is evaluating the solutions based on the fitness function having the best to survive fort the next generation. For each task, the fitness function counts the number of resource assigned to the team responsible to perform the task. The task time is then estimated based on the number of resources who are in the team assigned to it. The task that is under computation, in addition, should not be waiting another task to finish. The task that has a precede one, should wait till it finish. What is noteworthy that each team adds up each task time they work on to their working time, so the overall estimated project time is then the longest team time.

The algorithm named Kang01 starts by initializing the Simulated Annealing (SA) algorithm parameters of initial temperature, the initial solution, and the internal loops. The algorithm has two loops, where an internal one is designed to create solutions out of an old one using a function called PERTURB. If the new solution is better than the old one, then the solution will be accepted, or otherwise a probability will be attached to this solution decided upon the temperature and an

exponential function, having a linear relationship between probability and temperature. The temperature cools down when a solution by the fitness function is proved to be better by the internal loop. The cooling down has a factor that has also be set for the algorithm. The external loop on the other hand, keeps checking whether the fitness function value of the new solutions has not changed in the internal loops. The advised values set for this algorithm are 100 for initial temperature, 0.95 for the cooling factor, 500 for the internal loop, the internal loop control with 2000, and 8 for the external one. The initial solution by this algorithm is created using greedy algorithm. That means the project tasks is sorted from larger size to smaller, and with a continuous loop according to the number of resources, each task at a time will be assigned to the competent resource, till all the resources are assigned (end condition of the loop). By this solution the SA starts perturbing a new solution by exchanging resources with their participation percentages between the project tasks. The time of each task in the project is estimated by dividing the estimated effort over the participation percentages of all the resources assigned. However, in order to the participation percentage of a resource to be counted, his/her skillset should meet the ones required for that task, otherwise the resource should not be counted for that task. The project time in this algorithm is accordingly considered as the overall tasks' time i.e. the cumulative time of all the tasks.

The algorithm named DiPenta01 works similar to the tandem algorithm of Antoniol01 too, however, the solution is instead represented by a sophisticated vector chromosome structure. The vector combines within both the tasks queue order, and the team formation of resources representations. Each gene for the queueing order part represents a task and the allele value represents the position of that task in the queue order. On the other hand, each gene in the team distribution part represents a resource and the allele value represents the team number that this resource is assigned to it. This algorithm starts by initializing population P of size 50. The GA repeats by 250 generations three main processes, until a solution satisfy the objective function. The first process is selecting parents from P using roulette wheel selection. For the selected parents, the second process performs a single-point crossover with probability 0.7, and mutation using two elements exchange with probability 0.1, then stores these new solutions. The third process is evaluating the solutions based on the fitness function having the best to survive fort the next generation. The fitness function works first on decomposing the single structure chromosome into two representations of teams and task orders. For each task, the fitness function counts the number of resource assigned to the team responsible to perform the task. The task time is then estimated based on the number of resources who are in the team assigned to it. However, in order to the resource to be counted, his/her skillset should meet the ones required for each task that his/her team is assigned to, or otherwise the resource should not be counted for time estimate to

that task. The task that is under computation, in addition, should not be waiting another task to finish. The task that has a precede one, should wait till it finish. What is noteworthy that each team adds up each task time they work on to their working time, so the overall estimated project time is then the longest team time.

The algorithm named Minku01 starts by initializing population P of size 64, where the solution is represented by a 2-D matrix chromosome structure. The rows in this chromosome represents the resources and the columns represents the tasks. The value of each cell of the matrix represents a percentage that the resource will participate with between {0, 1/7, ..., 7/7}. The GA repeats by 79 generations three main processes, until a solution satisfy the objective function. The first process is selecting parents from P using Tournament selection. For the selected parents, the second process performs a single-point crossover with probability 0.75, and mutation using a single element bit-flip with probability 0.01, then stores these new solutions. The third process is evaluating the solutions based on the fitness function having the best to survive fort the next generation. The fitness function of this algorithm works by calculating the overall participation percentages of resources to each task. The task that is under computation, however, should not be waiting another task to finish. The task that has a precede one, should wait till it finish. The time for this task is estimated by dividing the estimated task effort over the overall participation of all the resources to this task. In order to the participation percentage of a resource to be counted, his/her skillset should meet the ones required for each task the resource is assigned to, otherwise the resource should not be counted for that task. The fitness function in addition checks whether each resource is working on more than one task at a time, if so, then the overall participation is normalized by the number of tasks that the resource is assigned to. The overall estimated project time is then calculated as the overall time of all the tasks in this project.

The algorithm named Park01 starts by initializing population P of size 100, where the solution is represented by a cell array chromosome structure. The gene cells in this chromosome represent the tasks, and the allele represents the resources assigned to the task. It is noteworthy that a task in this representation can have more than one resource assigned to it, as the type of this representation is cell array. The GA repeats by 400 generations three main processes, until a solution satisfy the objective function. The first process is selecting parents from P using Tournament selection. For the selected parents, the second process performs a uniform crossover with probability 1.0, and mutation using a single element random change with probability 0.05, then stores these new solutions. The third process is evaluating the solutions based on the fitness function having the best to survive fort the next generation. The fitness function of this algorithm works by simulating the project time day by day. That means the function repeats until all the

tasks of the project are finished. For each task available to be done, the overall productivity of resources who are assigned to it is calculated. However, in order to the productivity of a resource to be counted, his/her skillset should meet the ones required for each task that the resource is assigned to, or otherwise the resource will be counted with less productivity to that task. The time for this task is then estimated by dividing the estimated task effort over the overall productivity calculated of resources to this task. The task that is under computation, in addition, should not be waiting another task to finish. The task that has a precede one, should wait till it finish. The fitness function, in addition, checks whether each resource is working on more than one task at a time, if so, then the overall productivity of this resource is normalized by the number of tasks that (s)he is assigned to. Accordingly, the overall estimated project time is then calculated as the overall loops that the fitness function has performed.

2.5 Benchmarking, Datasets and Measurements

Benchmarking is a procedure whereby a set of experiments are conducted with the purpose of comparing the performance of alternative solutions [104]. In order to benchmark a solution and compare it to the rest of related ones, three components are mainly used by different approaches in different field of study as in [105, 106], which are a process for benchmark and comparison, benchmark dataset(s), and measures that are useful to distinguish between these solutions' performance and quality. Benchmarking has also been identified with three components by [104]. The first component identified by [104] is named "motivating comparison", which encompass technical comparison and research agenda. The second component identified by [104] is "task sample". This component is concerned with the representative solutions sample for benchmarking. It is noteworthy that the intention while identifying representatives should not be to include as many solutions as possible, but to select a representative set of these solutions [104]. Finally, the last component is the one that provides the benchmarking performance and quality measures.

2.5.1 Benchmark Process

Throughout the identification of the benchmarking process careful analysis should be made to fully understand what and which data and information can be extracted to distinguish between the alternative solutions. The work presented by [10] for benchmarking provides guidelines on how to approach and evaluate a SBSE solution with different algorithms. One of these guidelines is how to validate and benchmark an approach. This guideline consists of four benchmark baselines that a researcher can select one for the comparison. These baselines are random search

results, known solutions constructed by hand, desirable solution of how goodness it is compared to empirical data, or efficiency of solutions checks by repeated trials for having consistent good quality and more speedily solutions.

Limited number of benchmarking approaches can be found that compared and presented in particular the differences between various solutions for software problems as in [105, 106]. In [105], their aim was to evaluate and compare different task graph scheduling algorithms with respect to the processor's performance using a unified basis that allow variations in parameters. Their benchmarking process encompasses identification of a set of algorithms for evaluation, classification of algorithms, and suitable performance measures. The measures they have used are based on the parameters that have correlation with the processor performance. Their classification for the algorithms in addition was based on the number of processors required, the processors' network and structure, and computation cost characteristics. Their evaluation of the algorithms was performed using different datasets with different characteristics based on the classification, the set of algorithms they selected, and measurements they adopted. In [106] on the other hand, the aim was on evaluating classification models of software defect prediction. Their process for benchmarking involved identification of datasets to be used, outcome quality measurements, and classification of prediction models.

Both processes used by [105, 106] represent the same strategy that is applicable to be adopted in any other discipline. Therefore, our proposed benchmark approach adopted for the work carried out for this thesis follow this structure.

2.5.2 Problem and approach's classification

Three main studies in [5, 11, 17] have provided classification of SBSE problems. The work presented in [17] encompassed different software engineering optimization problems formed by two taxonomy perspectives. These perspectives are linked to software engineering and optimization. Both perspectives by [17] are considered as criteria to provide the aspects for a problem's taxonomy. The software engineering perspective involves development stages -such as requirement, design, etc.-, models -such as waterfall, and agile-, and the further description of a problem's subject. The optimization perspective on the other hand has the objective (fitness function), characterization – discrete, or continuous -, constraints, and the nature of the optimization problem in terms of polynomial and non-polynomial as P, NP, etc. As these aspects describes a larger area of SBSE than the problem presented in this thesis, the taxonomy proposed by [17] is not applicable for more specific problems of search-based software engineering as this thesis focuses on.

Different terminologies have been introduced by [11] for the approaches that solve software engineering problem using optimization techniques. One of these terminologies however, can

describe the work in this field as project planning in software management. Further work on this subject is provided by [5]. In their work, they have introduced a classification of subjects that belongs to software project planning and management. The work is similar to what this thesis evaluates and is classified in their study as minimizing software project completion time approaches. However, there is no classification by [5] for further extensions of minimizing the software project completion time problem. These extensions can be seen by the various attributes and parameters used in optimizing SSSP problem. Details on attributes and parameters used by different SSSP approaches can be found in [24]. In [24], they have provided a comprehensive study that demonstrate the attributes and aspects of the SSSP problem. Their findings have been presented in a qualitative sense to show what, and which criteria can be used to compare between SSSP approaches. They have used this criteria to demonstrate the possible approaches that are more suitable for adoption in the industry. Using their findings of attributes, parameters, and aspects of SSSP problem, the work on this thesis has been enhanced to create four categories of SSSP classification presented in Section CHAPTER 1.3.3.

2.5.3 Benchmark Measurements and Statistical Tests

There is only one study that has presented the most important measures to be used in particular for search-based software engineering approaches validation which is in [17]. This study categorized the validation measures into two separate groups. The first one discusses measures and metrics for fair comparison between different optimized approaches. In this group, four measures are recommended, which are the fitness value, the search time, arithmetic mean, and hit rate. In addition, they strongly advices to repeat the search of a single approach typically 30-50 times to capture the effects of random variations. These measures are adopted for the work on this thesis, and presented in Section CHAPTER 1.3.5. The second group involves the use of descriptive statistical analysis for central tendency and variability of results, and inferential statistics for accepting and rejecting hypothesises. They have advised the use of inferential statistical analysis with main concern about possibility of a Type I error i.e. concluding outperformance of an algorithm over another whereas in fact is not true. However as these statistical measures can demonstrate the feasibility and effectiveness of a new approach, they are not suitable to be used in our research while the arithmetic mean of different SSSP approaches are under investigation. In case of addressing stochastic nature and variation of performance between different sets of experiments they have referred to inferential statistics detailed in [107].

An important discussion of inferential statistical tests to assess randomized algorithms in software engineering is presented in [107]. In their discussion, measures are pointed out for their importance to plot the differences between randomized algorithms. These measures are T-Test and U-Test. Both measures are counted as statistical measures for parametric and non-parametric test respectively, to compare between two sample datasets with a hypothesis for testing whether

these data have distribution properties or not. The U-test for example, which has different names as Mann-Whitney-Wilcoxon, and rank-sum test, is a powerful statistical test, however, in our case of benchmarking SSSP approaches we are not testing the datasets rather to compare and evaluate the approaches' results.

Measures that have been used for benchmarking by the studies in [105, 106] are adjusted to their comparison subjects. In [105], they have used the algorithms' output, and computation time as a basic metrics for comparison. In addition, they have defined the upper and lower data boundaries and used that to represent the best solution, the arithmetic mean of each algorithm, number of processors used, and the normalized schedule length of each to solve the scheduling problem. Moreover, they have developed a scalability factor indicator to show how each algorithm can scale as the scheduling problem increases. In [106] on the other hand, they have used the hit rate to demonstrate whether each model is capable to provide a feasible solution on each runtime or not. Moreover, they have used Area Under the receiver operating Characteristics Curve (AUC) indicator for the classification models to identify whether each solution produced by each model is excellent, good, or worthless. In addition, they have used Nemenyi test recommended by [108]. This test is a non-parametric inferential test similar to Friedman test that is able to find differences for null hypothesis testing. These measures are also used by [109-111] too, as they are appropriate when conducting hypothesis tests for a single problem involving different optimization techniques, but not for a comparison of different SSSP approaches. It is obvious that part of their adopted measures such as hit rate, approaches output (fitness function output), computation time, and arithmetic mean are appropriate to demonstrate the differences between SSSP approaches and capable to provide evidence on performance to each. Therefore, these measures are used for the work carried out for this thesis detailed in Section CHAPTER 1.3.5.

Throughout the work carried out for this thesis, an accuracy measure has been found that can demonstrate the differences of accuracy between SSSP approaches. This measure is called Mean Arctangent Absolute Percentage Error (MAAPE) presented in [112]. As the Mean Absolute Percentage Error (MAPE) is well-known to its efficiency to forecast accuracy of methods, models, etc, [112], the work in [112] has provided an enhancement over MAPE using the Arctangent function to limit the outcome's boundaries. This measure has been used in the work carried out for this thesis and detailed in Section 43.6.6.

2.5.4 Available Repositories for Software Engineering Studies

Two basic repositories are available for software engineering research, the International Software Benchmarking Standards Group (ISBSG) and the (tera-PROMISE) of software engineering research data. However, neither of them contains valid datasets with useful information and data to be used for HRA problem [5]. Another repository is offered in "An Instance Generator for the

Project Scheduling Problem" that has been created and made available by the work presented in [14]. This repository generates datasets with a different number of resources, and tasks. Yet, these repositories offer datasets that have limited information for resource allocation inputs especially for resource information such as the skills and their associated productivity, as well as the interdependency between project tasks. In addition, the optimal solution and its fitness function value information are not offered by these repositories. The description and details of these repositories are reported in the following subsections.

International Software Benchmarking Standards Group (ISBSG)

The ISBSG is the first and most famous repository in software engineering. However, this repository is not freely accessible. Different types of research data are offered in this repository for software engineering research. This repository is mainly holding datasets that contains information about software projects. However, these projects can be organized into two categories. These categories are development projects, and maintenance projects. The information contained in the development projects datasets offer data that can be mainly used for the purpose of defects, development methodologies, software architecture, platforms and their relationship with effort estimation. The information regarding maintenance projects on the other hand are about the organization that the project belongs to, the application type and activities considered by the projects, size and effort estimation of each project, defects, platforms, hardware, and programming languages used by these software maintenance projects. However, important the datasets in this repositories are missing the attributes and parameters of SSSP problem.

Tera-PROMISE

The Tera-PROMISE repository offers a wide range of software engineering datasets. These datasets are mainly used by researchers on their previous research, and they have made these datasets available by this repository. This repository is established by [113]. The main contribution of this repository is the free access and availability of its datasets. These datasets are grouped into different categories according to the software engineering branches they belong to. These categories are code analysis, defect, dump, effort, green mining, bug issues, prediction models, requirements engineering, MSR, performance prediction, refactoring, search-based software engineering, social analysis, software aging, software maintenance, test generation, developer and project spreadsheet analysis, and other datasets that contains information regarding specific programming languages or platforms data. The main datasets and contributors are for defects, bug issues and effort estimation research purposes with 61, 35, and 14 datasets, respectively. However, important the datasets in this repositories are missing the attributes and parameters of SSSP problem.

An Instance Generator for the Project Scheduling Problem

This repository does not hold any datasets, but it provides an automatic dataset instance generator. It has been used and introduced by [14]. The generator provides data for different problem classes of project scheduling developed in Java. These classes belong to the parameters the project contains such as number of tasks, resources, and skills. These parameters are the information that the generator can provide for a new single problem instance. In addition, the information might contain data about dependencies between project tasks represented in term of task precedence graph. Moreover, salaries for each resource are also provided by this generator. However, in order for the generator to provide this information, a configuration file has to be used with it. This configuration file contains syntax that has the instructions of what parameters and attributes the results should include. The configurations of different problem instances are categorized based on the parameters the problem includes. The repository offers different files. These files control the number of tasks, resources, and skills of the instances. The ranges of tasks are 10, 20, and 30. The ranges of resource are 5, 10, and 15. The number of files for these configurations are 36.

2.6 Conclusion

Both studies in [5, 24] provide an important foundation of the search-based software project time minimization aspects, parameters, and features for evaluation. Throughout the reviewing of related studies, it was clear that both [5, 24] are focusing on the optimization techniques, variables (aspects), and data used to validate the approaches. These points are significant to address the issues regarding staffing and scheduling software projects. However, the approaches using the optimization techniques are also addressing an additional aspects such as the implementation of team and individual allocation as stages in the optimization process, as well as the constraints and penalties that are adopted in the case of constraints violation.

On the other hand, even if a discussion was made regarding applicability, usefulness, etc. of the proposed SSSP approaches, runtime and performance benchmarking of these solutions is also important to cover. In [24] they qualitatively claim that the solutions proposed by the approaches outperformed the experts' assignments. Yet, no comparison between the SSSP approaches and experts' solutions are detailed in these studies. Nonetheless, having in mind that the aim of optimizing HRA is to create a decision support not a decision making system stated by [5], more investigation on how the PMs perform their allocation to different allocation problems can provide evidence of the software organization projects' complexity and suitability of the proposed solutions.

Our project aims to evaluate the approaches not only by addressing their models, but also to test their robustness, accuracy, and precision through the implementation of these approaches. In addition, both [5, 24] studies do not consider a comparison between the expert method and the optimization approaches. Doing this can lead to a deep analysis and experimentations especially for the important aspects and variables that have to be considered by SSSP approaches from an industrial point of view.

It is also important to address how the SSSP approaches presented in [5, 24] have formalized their problem, provided their solution, and introduced their approach's validity. It is clear that lack of benchmarks for SSSP approaches of their capabilities in providing solution especially for different software project environments exists [5, 114]. From that sense, it is important to establish a foundation for a benchmarking approach that can provide evidence of each SSSP approach's performance, robustness, accuracy, and suitability. It is also important to capture how these approaches can adhere with industry practices and their capacity to provide solutions for different project problems and environments.

Chapter 3 Benchmarking Process for Staffing and Scheduling Software Projects Optimization Approaches

This chapter details our proposed approach for benchmarking and evaluating the SSSP approaches.

3.1. Introduction

Many approaches have been proposed in the last three decades tackling the problem of HRA in software projects. As one of the software engineering problems, it has been introduced to the software engineering community as part of the Search-Based Software Engineering (SBSE) by [10]. The SBSE term, however, includes all the areas that is benefiting the use of search-based algorithms to solve software engineering problems. In the work presented by [5], this problem is defined more precisely as one of the Search-Based Software Project Management (SBSPM) problems.

SPM is concerned with different management aspects including resource allocation. This thesis concentrates on HRA with consideration of software project time span minimization. This problem is introduced as software project completion time minimization by [5], and was illustrated as Staffing and Scheduling a Software Project (SSSP) addressed in Section 1.2. This problem, in particular, has received a widespread attention in SBSE and many SSSP approaches can be found that are tackling it. However, each approach provides a solution according to some or part of the software project properties. Therefore, these approaches need to be compared against each other, to capture any variances in their behaviour, and to measure their efficiency and suitability for industrial adoption. Accordingly, the use of benchmarking and statistical measures to challenge and compare between the SSSP approaches is required.

Benchmarking is one of the important areas that most of the search-based software engineering studies as in [5, 110, 114] have emphasised its importance. However as these studies have highlighted the absence of benchmarks due to a lack of software engineering repositories that encompass suitable datasets, benchmarking has been addressed as one of the their future work directions.

Our overall research framework for conducting the evaluation and benchmarking consists of four main stages. These stages are: problem definition, benchmarking, industrial settings evaluation, and interpretations of findings. These four research stages are described in the following Figure 9.

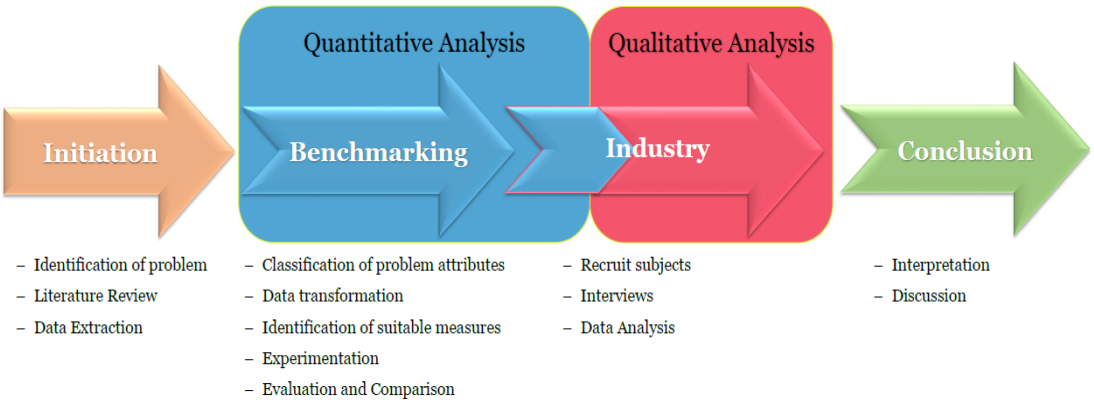

Figure 9: Research Framework

From Figure 9, it can be seen that our research starts with problem definition. To identify the problem within its context, and for best interpretation and formation of research question(s), a comprehensive literature review was performed. This review has led to identify the gaps in the research field, and determine the commonalities between the proposed SSSP solutions.

While many approaches have been proposed to solve SSSP problem, a gap in the research was found in demonstrating how they perform against each other. The major problem encountered in this regard was to identify a common definition that makes these approaches comparable. This has led to lookback into the optimization problem formalization that each aims to identify, and the common concepts they use. While some of these approaches, as in [14, 22], are expanding their problem to have multi-objectives or goals. The common objective shared among a large number of these approaches is tackling project time minimization. What is noteworthy is that these approaches have adopted a comparison between different optimization techniques to validate their solutions, and none has made a comparison between what they propose and others.

It is well-known that for evaluating an optimization approach, synthetic data can be used. SBSPM approaches mainly use simulated data to test their proposed solution, and limited number of approaches have validated their solution using industrial data. In our case, a dataset has to be made available for HRA optimization and software project time research. With the absence of any useful dataset, the first part of our research was to create a suitable one. This dataset is established based on archival project documents. The core of this data contains information on a small real-world software project from an international software company, where their name is kept anonymous upon their request. This work is depicted in Figure 9 by (Data extraction) of the first research stage.

Three main issues are subject to benchmark and evaluate in SSSP approaches. These issues are the allocation method proposed, the computation time, and the accuracy and precision of estimated project time span that these approaches can provide. On the other hand, each approach is subject to capture its suitability and capacity to adapt to different industrial settings problems. To achieve this, two main stages for benchmarking and evaluation, and industrial settings study are performed depicted by the blue and red arrows in Figure 9.

For the benchmarking and evaluation stage, we have concluded a process that can bridge the gap of demonstrating the differences between SSSP approaches. We start with the assumption of a set of SSSP approaches that all have to be tested for their capacity, capability, accuracy, and stability in finding optimal or near optimal solutions. For this purpose, the first artefact of benchmark dataset will be used. However, as each of the proposed SSSP approaches has a different interest and use of the SSSP problem parameters and inputs, we have defined four classes to demonstrate their use of problem's parameters and inputs as complexity levels, and composed them into different datasets. These datasets combine five levels of information complexity corresponding to the classification made on the SSSP approaches based on the problem inputs proposed by each. This work is depicted in Figure 9 by both classification of problem attributes, and data transformation steps.

The benchmarking stage, in addition, provides a process to identify what approaches can be selected, how to perform a comparison, and what measures and criteria can be used. Therefore, statistical measures and quality metrics are adopted from different comparisons and evaluation studies as in [15, 17, 107] such as hit rate, arithmetic mean, standard deviation, etc. as well as measures that can provide accuracy forecasting such as mean arctangent absolute percentage error [112]. By using the benchmark process, significant findings and differences between the

SSSP approaches were identified. Details and findings for benchmarking and evaluation of SSSP approaches are presented in Chapter 4.

On the other hand, to cover-up how and why these approaches might hold within drawbacks to the application within the software industry, a supplementary qualitative research has to be involved as pointed out by [115]. Software project time span minimization research also requires an understanding of the current practices of resource allocation in the software industry. To better assess these points, an industrial settings study was performed. This study however, involved a mixed-methods approach, depicted by the blue and red boxes in Figure 9.

The purpose of using mixed-methods approach is to quantitatively evaluate the project managers' solutions for the SSSP complexity levels, as well as to capture the different aspects and practices of resource allocation they adopt and use. Based on the results and findings from the industrial settings stage combined with the results from the benchmarking and evaluation stage, the interpretation is the last activity to be performed in our research framework. This part of the research stage concludes the overall findings, trends for future, and its limitations.

The remainder of this chapter provides a systematic comparison and benchmarking process suitable for SSSP problem in Section 3.2, proposed classifications of SSSP approaches in Section 3.3. In addition, this chapter details the datasets, their complexity levels, and the optimal solution of each level in Section 3.4, a set of measures for benchmarking SSSP approaches in Section 3.5, and summarize the overall benchmarking process in Section 3.6.

3.2. A Systematic Approach for Comparing SSSP Approaches

Comparing between the approaches proposed for SSSP problem requires a systematic process that clarify their outcomes and resulting in reliable comparisons. Our proposed process for performing systematic and reproducible performance comparison of SSSP approaches consists of sequence of steps combined with evaluation datasets and a suite of quality measures on which the SSSP approaches can be compared. The proposed workflow for evaluating a set of SSSP approaches consists of the following steps:

1. Select a set of candidate SSSP approaches that are capable of solving a resource allocation problem and belong to the same class – see Section 3.3 -.

2. Select the suitable dataset from the benchmark dataset that belong to the same class of approaches selected containing the desired resource and project properties (e.g. skills, task dependencies, etc.)-See Section 3.4-.
3. Run each approach for the configured dataset for a substantial number of times, (e.g 100 times).
4. Record for each run the result of estimated project time, and the computation time of that run.
5. Compile the results and measure their performance using the benchmark metric suite -see Section 3.5-.
6. Rank the candidate SSSP approaches based on their score in the overall quality measures -see Section 3.5-.

These steps are depicted in the following Figure 10. As can be seen in Figure 10, after identifying the approaches and the classes that they belong to, and selecting the suitable benchmark dataset, the datasets located on the left down of the figure is fed into each approach. As most of the approaches perform heuristic optimization using a probabilistic optimizer, the next step in the benchmark process is to perform multiple runs for each of those approaches. Different number of experiment runs are used by the SSSP approaches. Some of these approaches have used 30 runs such as [22, 94], whereas others used 100 as in [15]. The rule of thumb is to use 30 runs defined by [107]. Due to the stochastic process of the optimization techniques the number of experiment runs can be 30-50 according to [17]. However, the number of runs of each approach in our experimentations were set to 100. The reason for that is to widely investigate and accurately depict the range of possible results, and to form a better picture of SSSP approach analysis [107]. This reasonable number of runs is carried out to properly analyse the behaviour of the approaches. Recorded results of the approaches for each run are then used for the evaluation and comparison step depicted in the middle of Figure 10.

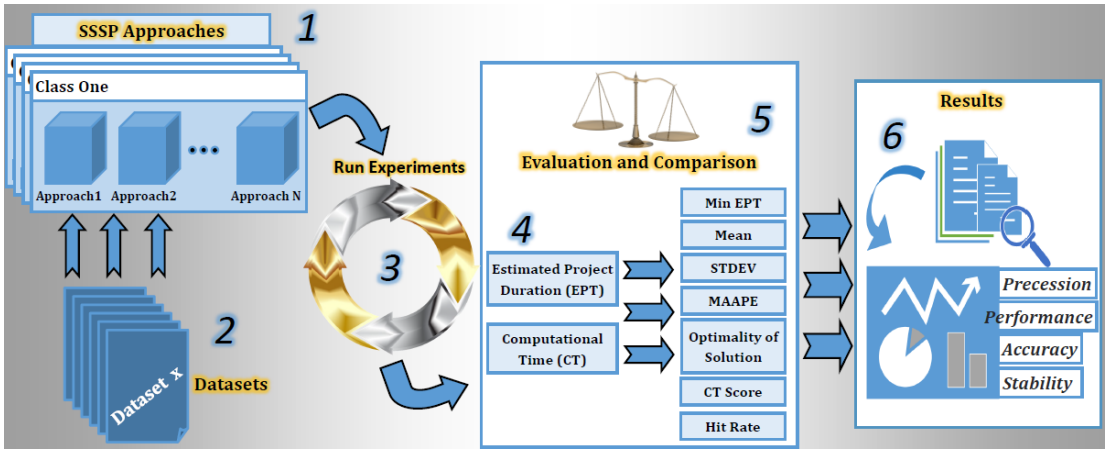

Figure 10: Proposed Benchmarking Approach

The main purpose of using heuristic techniques is to find the best solution in the least possible time. This depicts two outputs of the optimization defined in the evaluation and comparison step, which are the main concern in this study. These outputs are the Estimated Project Time (EPT) and the Computation Time (CT). EPT represents the best solution ever found in each run, which is the optimal output solution by the approach retrieved from the value of the fitness function. On the other hand, the amount of effort the approach expended in finding its best solution is represented by CT, which is the time consumed by the system to find that optimal output solution value. While the EPT is important to demonstrate the accuracy of the approaches, CT is also important to consider when performing a comparison between SSSP approaches [17].

After running the experiment 100 times on each approach, the results of estimated project time and computation time are stored, seven measures are proposed to be used in the benchmark through the evaluation and comparison step to depict the quality and differences between the approaches. The description and demonstration of these measures can be found in Section 3.5 of this chapter. The choice for these measures is motivated by the fact that they are seen as the most useful way to represent effectiveness and performance amongst the approaches [105]. These measures are anticipated to depict the accuracy, precision, performance, and stability of the approaches' results. Stability and Precision of the approaches' outcomes can be depicted by Standard Deviation. The performance of an approach in addition can be depicted by the computation time that shows the speed of an approach to produce a result. Accuracy of the approaches' outcomes on the other hand can be depicted by two measures. The first one is the mean arctangent absolute percentage error (MAAPE). This measure depicts how far the outcomes are from the actual optimal solution, which accordingly will demonstrate how accurate the

approach is. The second measure is the optimality of solution. We have defined this measure to plot the accuracy of each approach according to the class that it belongs to. Those two measures of accuracy require the optimal estimated project time value to be defined for each level of dataset complexity. These values have been manually estimated as an optimized (optimal) solution and are provided within our dataset including the derived solution for each level described in Section 4.53.4.2.

3.3. Classification of SSSP Approaches

During the literature review, part of our work was on identifying the parameters and attributes defined within the SSSP approaches. The idea behind that combines two main reasons. The first reason is to find whether these approaches have defined a common problem concept that makes them comparable. The second reason is to identify common attributes and parameters used as input to these problems with the aim to address any differences, and if any, the next step is to classify them into different comparable groups.

In this study, the findings conclude that the mainstream of SSSP approaches are focusing on the number of resources available to the project, their skills related to the ones required for the project, and the estimated effort for each project task with consideration of precedence relationships between these tasks as in [14, 15, 18]. However, there are approaches that employ simulation techniques such as the queueing system to formalize the allocation problem into a simulation systems with the purpose of researching and understanding the outcomes, and their relativeness to the real-world problem as in [20, 21]. The resource allocation problem is still open for this type of application, yet the approaches employing these techniques require less project and resource information, and measures to research the outcomes. It is also important to notice that skills and dependency relationship are used as constraints by many SSSP approaches in their problem definition. That leads us to recognize whether the constraints they use are hard or soft. Both types are used within SSSP approaches and that leads us to separate these approaches into different classes. For example, approaches as in [22, 94] use soft precedence relationship constraint. This is due to the nature of their fitness function. This function automatically deals with any precedence relationship in a way that delays the task that has that relation till its predecessor is finished. Therefore, some SSSP approaches use only data about the number of resources available and estimated effort of each task, whereas others use more sophisticated inputs. Classifying these approaches based on project and resource's attributes, and according to

the optimization problem constraints presented by the SSSP approaches shows four classes, which are:

Table 11: SSSP Classes

SSSP Classes	Class One	Class Two	Class Three	Class Four
SSSP Features				
Estimated Effort	✓	✓	✓	✓
Number of Resources	✓	✓	✓	✓
Project Task Dependency		✓		✓
Software Development Skills			✓	✓

• **Class One**. This class contains the approaches that require inputs only of estimated effort of project tasks and the number and productivity of human resources.

• **Class Two**. This class contains the approaches that require inputs of estimated effort of project tasks, dependencies between these tasks, and number and productivity of human resources

• **Class Three**. This class contains the approaches that require inputs of estimated effort of project tasks, skills required for each task, and number, skills, and productivity of human resources

• **Class Four**. This class contains the approaches that require inputs of estimated effort of project tasks, dependencies between these tasks, skills required for each task, and the number, skills, and productivity of human resources.

These classes can be seen as a taxonomy of SSSP approaches, where some can possibly be part of multiple classes as they are able to determine the optimal allocation of resources for simple as well as complex SSSP problems. When benchmarking SSSP approaches, it is critical to note that proposed approaches generally solve different variations of the resource allocation problem, taking into account different parameters, such as worker skills, or tasks dependencies. To evaluate the relative performance of SSSP approaches they need to be applied to the same problem with the exact same inputs, which is why we propose to group SSSP approaches into classes according to the inputs and constraints required by each. For a complete survey on SSSP approaches' optimization parameters, and input attributes the reader can refer to [5, 17, 24]. The benchmark data follows this classification as it defines optimization challenges within these four distinct

classes to facilitate the uniform comparison of SSSP approaches. The detail of each dataset is presented in the next section.

3.4. Benchmark Dataset

The first artefact in this thesis for benchmarking is a flexible and configurable dataset. The dataset is a small real-world data from an international software company and holds information regarding both software project and human resources used to develop that software. This data includes information about eight components of the software projects, and twelve human resources were available to that project assigned to complete it. The project represented in the dataset has an estimated time using COCOMO [98]. The time estimated with those resources available was 75.16 days, with an estimated Man-Day equal to 964.

While there is a diversity of approaches each employs different attributes of project information to solve the allocation problem of SSSP, any additional parameter added to the simple input information of estimated project tasks effort and the number of resources is counted as a level of input complexity. Based on this definition and corresponding to the classification described in Section 3.3, the dataset is composed of five complexity levels. These levels describe resource allocation problems of increasing complexity and parameters. Accordingly, each level represents a dataset that holds part of the original project data provided by the contributor organization. The optimal solution for each one of these levels (referred to as min value) as well as the worst-case solution values (referred to as max value) are defined. Section 3.4.2 details the dataset used in this thesis.

The inputs required for resource allocation can be the estimated effort of project tasks, task dependencies, skills, and/or resource productivity. Each one of these inputs is represented in the dataset by numbers except the skills. Skills required for developing each task or offered by a resource are representing languages and technologies, and represented in the dataset using the name of this language or technology such as java, or UML. Estimated effort of each task is represented by person-day. Each task in the dataset has the value of dependency attribute represented as the task number that the task is depends on. The project tasks in the dataset are named Work Packages (WP) for the unity of definition as it is used for industrial settings study. Productivity of a resource is represented by the same metric used by [28]. A resource can be productive as a normal person, which is equal to 1, less than a normal person represented by a value less than 1, or twice the normal person represented by 2. The description of the dataset

levels, their resource allocation problem attributes, and the input values of each are introduced in the following sections.

3.4.1 Dataset Complexity Levels

For benchmarking and challenging the performance and applicability of SSSP approaches, five levels of problem input complexity are proposed. These levels are made as case studies representing the level(s) that a SSSP approach is capable of solving. The first two levels have the same concept of sharing the same productivity among the resources. That implies all the resources are equally productive and can perform any task regardless of the skills and competencies that the task requires and that the resource possesses. The next three levels however are different from the first two, since they consider the resources' productivity according to the skill(s) required to develop the project tasks. Moreover, level three and four have productivity of a resource either one (1) or zero point one (0.1). These values are defined according to the nature of time equation used by SSSP approaches. If we kept the concept of one or zero when a case where a resource does not possess the required skill(s), then his/her productivity will provide undefined value to the estimated effort for the task that (s)he is assigned to and causing the experiments to fail. In addition, level five has the values of productivity for each resource ranges between zero point one (0.1) to four (4) representing each skills (s)he possesses. The detailed description of each level is as follow:

- The first level represents the simple resource allocation problem that has two type of input data. The first one is about the tasks and the estimated effort of each. The second input data represent the number of resources available to perform these tasks. Productivity in this level however, is set to be one.
- The second level of this dataset has three types of resource allocation input data. The first represent the number of available resources, the second is the estimated effort of each task, and the third one represents the task dependencies in which the value of this input to each task represents the task(s) that it depends on.
- The third level in this dataset has the number of available resources, and the estimated effort of each task similar to the previous ones. However, it has also the information about skill(s) that each task in the project requires, and each of the available resources possesses. The values of this type of input can be either 1 or 0.1.
- The fourth level of this dataset has four types of allocation input data. Similar to the previous ones the first two inputs hold the information regarding the number of available resources, and the estimated effort of each task. Moreover, another input holds the

information regarding the task(s) that this task is depends on. The last input information is the information about the skill(s) that each task requires, and each of the available resources possesses. The values of productivity for this level of input are also either 1 or 0.1.

- Similar to the fourth level, the fifth one in this dataset has four allocation input data. However, the only difference between the fourth and the fifth is the information about the skill(s) that each task in the project requires, and the ones possessed by the available resources. This information represents their productivity regarding each skill and is represented in the dataset by a range from zero point one (0.1) to four (4).

3.4.2 Resource Allocation Scenarios of Dataset Complexity Levels

According to the complexity levels defined in the earlier section, five resource allocation problem scenarios are created. These scenarios are constructed based on the SSSP classification. The first four scenarios follow the description of the four classes of SSSP classification. However, the maximum productivity of a resource in these levels is no more than one. That means productivity of a resource can only be as a normal person (producing the same amount of work expected by 8 hours working time). Unlike these levels, level five description of resource's productivity can vary from 0.1 to 4. This level adds to the problem complexity the variability of productivity between the resources who shares the same skills. The response of the approach handling this accumulation of productivity, dependencies, and skills can demonstrate its effectiveness. The overall reason for these different scenarios is to capture the behaviour of the approaches in terms of EPT and CT while challenging them with the increasing level of information.

In addition, these scenarios are used in our industrial settings evaluation study. This study was to capture the similarity between what the datasets provide and the current business problem. The subjects in this study were asked to perform an allocation to each one of these scenarios, and their responses were recorded. The first part of the questions provided to the subjects were exactly the same following scenarios. The main intended establishment from this study is to validate these datasets. A hint is given at the end of each scenario description that sharing developers across WPs is not allowed. It is that the resource sharing counted in these scenarios as an unacceptable solution. The intention of this hint is to look like a tricky point which should guide the subjects to a good solution. It is important to take into account that you cannot use all the resources together doing all the activities at the same time. For instance, if we have two resources and five activities, we cannot use both resources to do the five activities at the same time without considering the negative impacts on their productivity. The description of the five scenarios, their constraints, optimal solution that we have manually estimated, and the optimal project schedule according to

this estimation are depicted in the following subsections. The context of these scenarios was designed to be a question for the industrial settings assessments.

Scenario 1:

The software development company Xee specialises in project-based software development and employs 12 developers. Project managers are responsible for staffing a project based on a variety of project and developer parameters. We would like you to consider the following staffing scenarios and provide the best project time span estimation. Xee has secured a new project which needs to be staffed. To this purpose the project manager has identified eight WPs to which developers need to be assigned. For this first scenario we assume that all developers are completely uniform, i.e. they have the same skill set and have comparable productivity. In addition, for each of the eight WPs the required effort has been estimated in terms of Person-day as follow:

- WP1: 82 Person-day
- WP2: 223 Person-day
- WP3: 180 Person-day
- WP4: 132 Person-day
- WP5: 190 Person-day
- WP6: 50 Person-day
- WP7: 62 Person-day
- WP8: 45 Person-day

As a project manager using this information, you are asked to perform an allocation that assigns the developers to WPs on this project while satisfying the following constraints:

- The project has to be completed as soon as possible
- Sharing developers across WPs is not allowed

Answer:

The answer for this scenario can be as follow. All those resources available can be used to form teams. While different alternatives can be made by different team formation, having them all in a single team to perform the project's WPs for this particular problem works as the optimal one. Therefore, we have a single team that consists of twelve resources, where each possesses the same productivity. In this case we can either make the team work on the WPs simultaneously, or they can work on these WPs sequentially. The former will not give the same result as the latter. If we made them simultaneously perform the project WPs then this will mean their productivity will be

divided by the number of these simultaneous WPs. That is the estimated effort for each WP is then should be divided by 1.5. The overall estimated project time then is approximately equal to 642.67 Days. The latter however will hit the target, which propose the development to work sequentially. The result from this way is as follow. 82/12 + 223/12 + 180/12 + 132/12 + 190/12 + 50/12 + 62/12 + 45/12 ≈ **80.33 Days**. The solution according to this way is depicted in the following Figure 11 by the project Gantt chart.

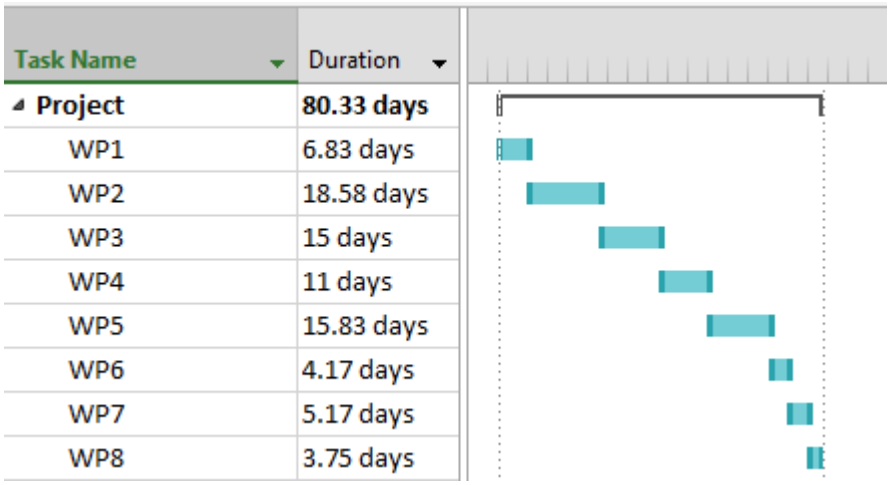

Figure 11: Scenario 1 schedule Solution

In Figure 11, it is clear that the development is done sequentially providing a waterfall development. The name WP however, is used to illustrate the different type of works required behind these WPs. In software development four major activities - analysis, design, coding, and testing - have to be performed during the development of the software. Therefore, these WPs requires that different activities be performed in order to complete each WP.

Scenario 2:

The second scenario we would like you to consider is similar to the first one but the WPs of the project now have dependencies, meaning some WPs have to be completed before others can be started. These dependencies are displayed in Table 12, and depicted by Figure 12 for more clarification. From this table, it can be seen that for example WP4 requires 132 Person-day to complete and cannot be started unless WPs 2, and 3 are finished. For this project the same developers are available as scenario 1. As before, perform an allocation and assign developers to WPs under the following conditions:

- The project has to be completed as soon as possible
- Sharing developers across WPs is not allowed

Table 12: Scenario 2 Project Attributes

Project		
	Dependency	***Workload***
WP1	-	82
WP2	1+3	223
WP3	-	180
WP4	2+3	132
WP5	4+6+7	190
WP6	4	50
WP7	3	62
WP8	7	45

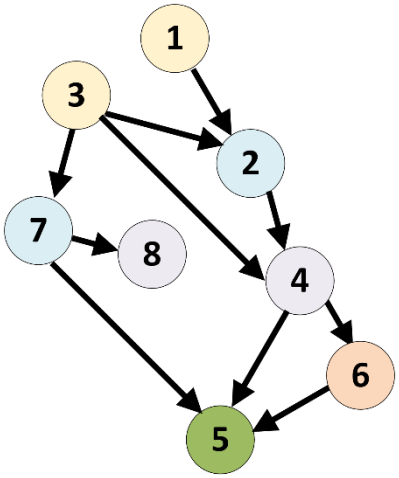

Figure 12: Level 2 Dependency Graph

Answer:

Again this problem can be solved having all the resources available formed in a single team. However, this team will work on project WPs according to their precedence constraints. Accordingly, their work will start by doing the jobs sequentially as follow WP1, WP3, WP2, WP4, WP6, WP7, WP8, and then WP5. By doing so, the project time will be exactly the same as for the previous scenario, which is equal to **80.33 Days**. This solution is depicted in the following project Gantt chart Figure 13.

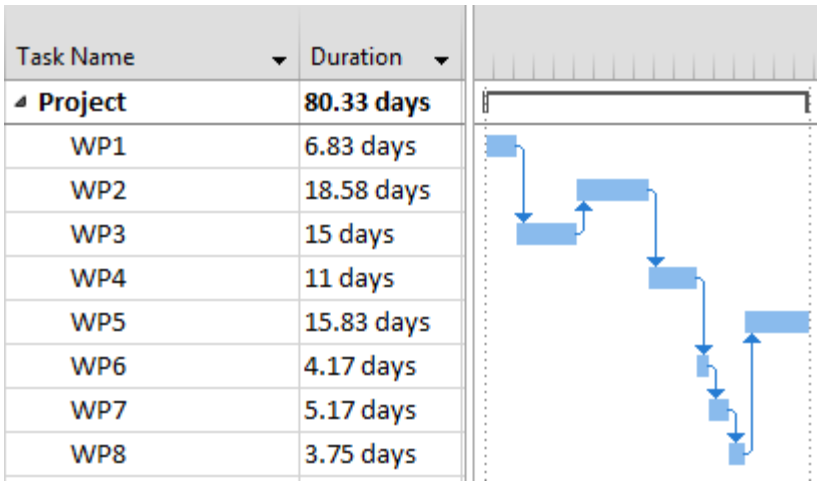

Figure 13: Scenario 2 Schedule Solution

From Figure 13, it can be seen that project WPs have the same estimated time as before according to the allocation of a single team. For instance, WP3 in this scenario has an estimated time of 15 days matching the same value for this WP in the previous scenario. Although this scenario provides dependency constraints between the WPs, forming the work sequentially has led to the same results.

Scenario 3:

In this scenario we would like you to consider developer skills against the required ones for project WPs. The following Table 13 lists the skill requirements for each WP in the project:

Table 13: Scenario 3 Project Attributes

Project		
	Skills	***Workload***
WP1	SQL, JDBC	82
WP2	SQL, JDBC	223
WP3	J2EE, Web	180
WP4	J2EE, Web	132
WP5	J2EE, Client Server	190
WP6	J2EE, Client Server	50
WP7	Java Networking	62
WP8	Java Networking	45

In Table 13, it can be seen that, for example, WP5 requires 190 Person-day and J2EE and client server skills. Moreover, dependencies in this scenario have not been considered. In addition, Xee has collected information on the skills possessed by its developers as well as their productivity. This information is listed in the following Table 14.

Table 14: Scenario 3 Resource Attributes

Resource NO	***Skills***	***Productivity***
1	Java Networking	1
2	Java Networking	1
3	J2EE, Web	1
4	SQL, JDBC	1
5	J2EE, Client Server	1
6	J2EE, Web	1
7	SQL, JDBC	1
8	J2EE, Client Server	1
9	SQL, JDBC	1
10	J2EE, Web	1
11	J2EE, Client Server	1
12	Java Networking	1

In this table the developer productivity is treated for all as of a normal person. Productivity indicators are provided for the corresponding skills. So, for example resource7 possesses SQL, with JDBC skills which makes him/her productive as normal person, and in addition, if the resource is assigned to a WP that requires different skill(s), then its productivity will be reduced to 0.10. According to this information you are asked to perform an allocation that assigns the developers to WPs on this project under the following conditions:

- The project has to be completed as soon as possible
- Sharing developers across WPs is not allowed

Answer:

The answer for this problem is quite simple too as for the previous ones. While the concern is about the skills, this means forming the teams can be according to these skills, so each team has the resources who are productive for the particular skill required for the WPs. Therefore, four teams have to be formed, where the resources in each possess specific skills. According to this allocation concept project time can be illustrated as the maximum working days among the teams. The formation of each team can be depicted as follow. Team one should have resources 4, 7 and 9. Team two should have resources 3, 6 and 10. Team three should have resource 5, 8 and 11. Finally, team four should have resource 1, 2 and 12. Team one will work on WP1, and WP2. Team two will work on WP3, and WP4. Team three will work on WP5, and WP6. Team four will work on WP7, and WP8. The maximum working days amongst those teams is team two i.e. team two is the one that works more than any other team and project time can be counted according to their

working days, which equals to **104 Days**. The solution for this scenario is depicted in the following project Gantt chart in Figure 14.

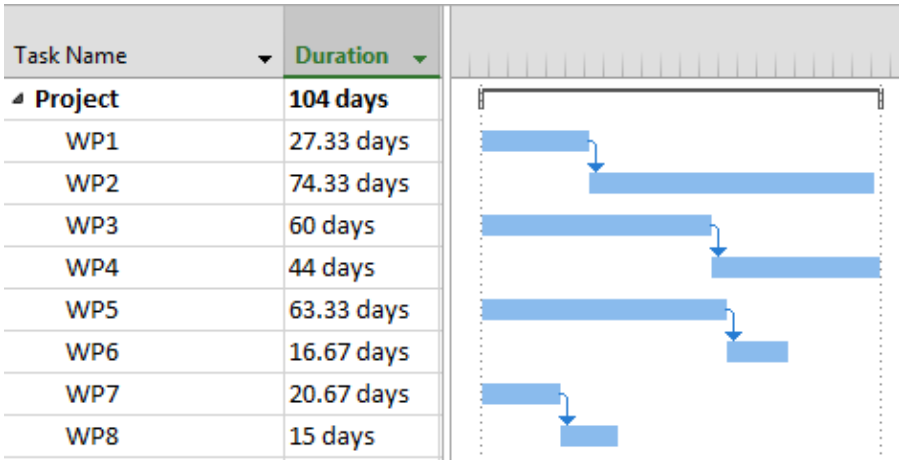

Figure 14: Scenario 3 Schedule Solution

From Figure 14, it can be seen that the project time can be defined as the maximum among the teams' work time. The maximum time across the teams is the second one. This second team who works on WPs three and four will requires 60 plus 44 days, which equal to 104 Days to finish their work. Accordingly the estimated project time is then equal to 104 Days.

Scenario 4:

In this scenario, we would like you to consider in addition to dependencies between WPs, the developer skills and productivity into account. The following Table 15 lists the skill and dependency requirements for the WPs of the project:

Table 15: Scenario 4 Project Attributes

Project			
	Skills	***Dependency***	***Workload***
WP1	SQL, JDBC	-	82
WP2	SQL, JDBC	1+3	223
WP3	J2EE, Web	-	180
WP4	J2EE, Web	2+3	132
WP5	J2EE, Client Server	4+6+7	190
WP6	J2EE, Client Server	4	50
WP7	Java Networking	3	62
WP8	Java Networking	7	45

In this table, it can be seen that for example WP2 requires 223 Person-day, SQL and JDBC skills. Moreover, this WP cannot be started unless WPs 1, and 3 are finished. In addition, Xee has collected information on the skills possessed by its developers as well as their productivity. This information is listed in the following Table 16.

Table 16: Scenario 4 Resource Attributes

Resource NO	Java Networking	J2EE, Web	SQL, JDBC	J2EE, Client Server
1	1	0.1	0.1	0.1
2	1	0.1	0.1	0.1
3	0.1	1	0.1	0.1
4	0.1	0.1	1	0.1
5	0.1	0.1	0.1	1
6	0.1	1	0.1	0.1
7	0.1	0.1	1	0.1
8	0.1	0.1	0.1	1
9	0.1	0.1	1	0.1
10	0.1	1	0.1	0.1
11	0.1	0.1	0.1	1
12	1	0.1	0.1	0.1

In this table for each developer their productivity is expressed in terms of a normal productive person, where a normal person productive for 8 hours a day. This productivity provides indicators for four different types of skills in the skills column. So, for example resource 7 possesses SQL, with JDBC skills which makes him/her productive on average as normal person, and in addition, if this resource is assigned to a WP that requires different skill(s), then its productivity will be reduced to 0.10. According to this information you are asked to perform an allocation that assigns the developers to WPs on this project under the following conditions:

- The project has to be completed as soon as possible
- Sharing developers across WPs is not allowed

Answer:

For this particular problem sharing resources might be practical, however, their productivity needs to be normalized according to the number of WPs they work on at the same time. Bear in mind that doing more than one WP at the same time implies reduction of the resource productivity. Then managing the resource to do a single WP at a time can lead to better solution. Having this in mind, the optimal allocation solution for this problem will be as follow. Resources 4, and 7 will perform WP1. Resources 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12 will perform WP2. Resources 1, 2, 3, 5, 6, 8, 9, 10, 11, and 12 will perform WP3. Resources 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12

will perform WP4. Resources 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12 will perform WP5. Resources 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12 will perform WP6. Finally, resources 1, and 2 will perform WP7 and afterwards WP8.

This assignment of resources can be seen as a dynamic team formation. Different teams in this particular allocation have been formed during the project development. We have seen that a team formed by two resources - as the one allocated to WP one – for another WP has been transformed into a larger team - as the assignment for WP two - that includes those two resources as well as resources 3, 5, 6, 8, 9, 10, 11, 12. It is well known that productivity of a resource working in different teams can be affected for example by communication overhead [6]. However, this form of allocation is not far from reality and can be seen in the current industry practice [9].

This allocation with consideration to the constraints of skills and dependency between the WPs for this scenario shows that the estimated project time is equal to 204.31 Days. The project schedule including estimated project time and the time for each WP are depicted in the next Figure 15.

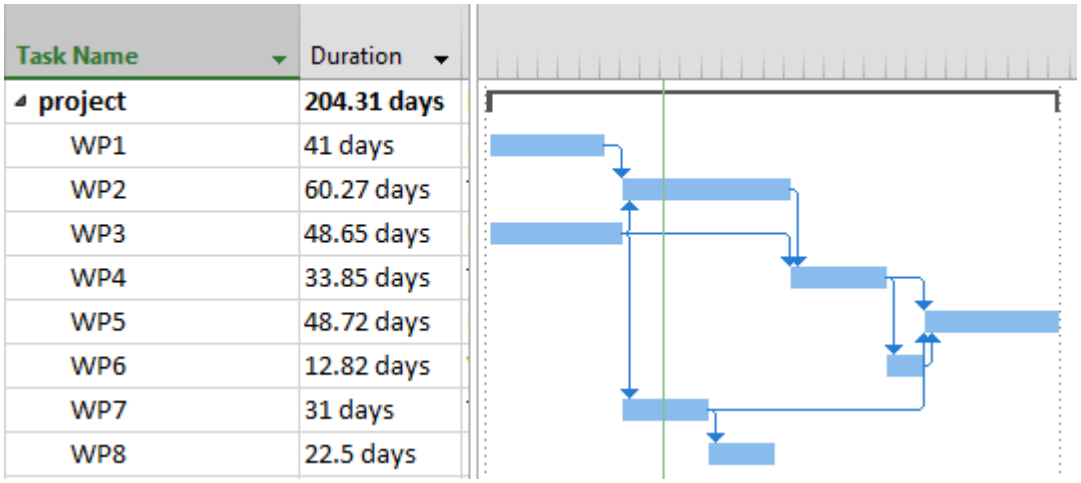

Figure 15: Scenario 4 Schedule Solution

From Figure 15, it can be seen that different WPs are simultaneously performed however, none of the resources assigned to these WPs are working on more than one at the same time. This makes the resources work to their full productivity, so project time can be the least possible. It can be seen from Figure 15 too that the project time frame defined by the critical path starts with WPs 3, 2, 4, 6, and then 5. This path conforms to the dependency constraint however, shortening the time of these WPs requires attention to their simultaneous ones as WP 1 with WP3, and WP2 with WPs 7, and 8. That means increasing or reducing the number of resource on these WPs might result in low quality outcomes.

Scenario 5:

In this scenario, we would like you to consider in addition to dependencies between WPs the developer skills and productivity into account. The following Table 17 lists the skill and dependency requirements for each WP in the project:

Table 17: Scenario 5 Project Attributes

Project			
	Skills	***Dependency***	***Workload***
WP1	SQL, JDBC	-	82
WP2	SQL, JDBC	1+3	223
WP3	J2EE, Web	-	180
WP4	J2EE, Web	2+3	132
WP5	J2EE, Client Server	4+6+7	190
WP6	J2EE, Client Server	4	50
WP7	Java Networking	3	62
WP8	Java Networking	7	45

In Table 17, it can be seen that for example WP6 requires 50 Person-day, and J2EE and client server skills. Moreover, this WP cannot be started unless WP 4 is finished. In addition, Xee has collected information on the skills possessed by its developers as well as their productivity. This information is listed in the following Table 18.

Table 18: Scenario 5 Resource Attributes

Resource NO	***Java Networking***	***J2EE, Web***	***SQL, JDBC***	***J2EE, Client Server***
1	2.5	0.1	0.1	0.1
2	2.75	0.1	0.1	0.1
3	0.1	2.25	0.1	0.1
4	0.1	0.1	2	0.1
5	0.1	0.1	0.1	1.75
6	0.1	2.5	0.1	0.1
7	0.1	0.1	2.25	0.1
8	0.1	0.1	0.1	3
9	0.1	0.1	1.5	0.1
10	0.1	1.5	0.1	0.1
11	0.1	0.1	0.1	1.5
12	1.5	0.1	0.1	0.1

In Table 18, each developer productivity can range between 0.1 to 4. This productivity provides indicators for four different types of skills in the skills column. So, for example resource 7 possesses SQL, with JDBC skills which makes him/her productive of 2.25 times of a normal

person, and in addition, if this resource is assigned to a WP that requires different skill(s), then his/her productivity will be reduced to 0.10. According to this information, you are asked to perform an allocation that assigns the developers to WPs on this project under the following conditions:

- The project has to be completed as soon as possible
- Sharing developers across WPs is not allowed

Answer:

Similar to scenario 4, the optimal solution for this problem is also by considering the dynamic team formation. However, productivity of the resources this time differ from one to another even when they share the same skills. Having this in mind, the same assignment of resources for scenario 4 is still the optimal one for this problem. The estimated project time for this assignment solution is equal to 112.49 Days. The schedule represented by the Gantt chart for this problem including the estimated time for each WP as well as the overall project time are depicted in the following Figure 16.

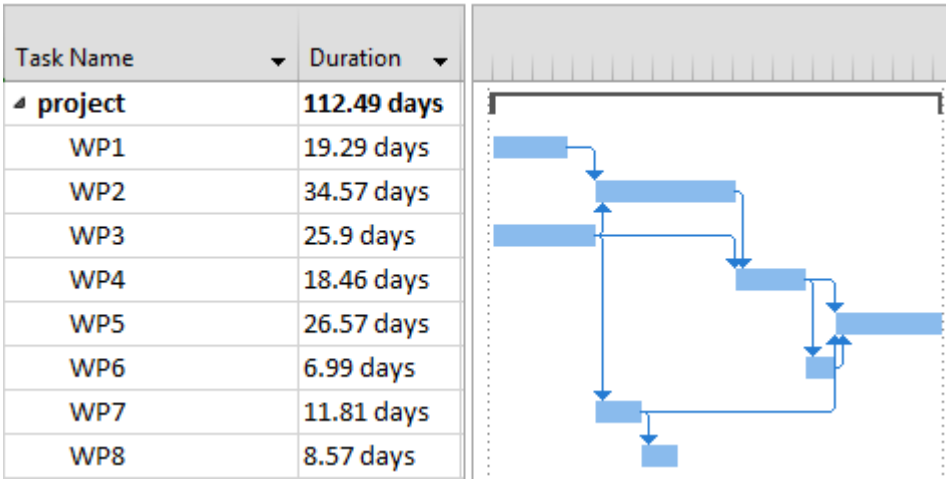

Figure 16: Scenario 5 Schedule Solution

From Figure 16, it can be seen that the same schedule pattern is performed according to the dependency constraints between the WPs. However, considering productivity of resources assigned to these WPs has shown that the time can be reduced for each WP as well as for the overall project. For instance, having the productivity information of resources 4 and 7 who are working on WP1 in this scenario managed to reduce the estimated time span of this WP from 41 Days in previous scenario, to 19.29 Days. Moreover, productivity in this scenario for resources 1

and 2 who are working on WP 7 has managed to reduce the WP time span from 31 Days in the previous scenario to 11.81 Days.

The five scenarios described earlier are the sets of data used for this thesis to challenge the accuracy and performance of SSSP approaches and the industrial settings study subjects with five levels of complexity. The description of these scenarios depicted earlier is to provide the reader with a context and an optimized (optimal) solution for each level. The reason behind picking this particular small project information for the work carried out for this thesis is due to the industrial settings assessment. As a set of Project Managers (PM) are the subjects for this study where their availability and time constraints should be considered, this has led us to make the datasets as simple and as easy to understand as possible for the PMs to find a solution especially for the extended levels.

3.5. Quality Metrics and Comparison Measurements

Measures form the modern quantitative judgement for evaluating and reporting the appropriateness of different models, approaches, and algorithms. As SSSP approaches in this thesis are subject for evaluation, different measures should be employed to evaluate their performance and quality against each other. Many studies have proposed measures that are suitable to capture the differences of approaches that solve particular software engineering problem as in [15, 17, 22, 107, 115]. As each of these studies provides a set of measures for a particular SE problem approaches, some of their measures cannot be employed in our evaluation study. Therefore, we should limit these measures based on their appropriateness and suitability to the evaluation criteria, and how effectively they provide information about post-sample accuracy [116]. Evaluating the SSSP approaches using different measures does not implies searching for the best, because no single one can be designated as the best. However, emphasis can be made to exclude any approach that performs badly.

The performance of a SSSP approach is usually measured in terms of optimality, i.e. how close the approach gets to the true optimal solution [105]. However, this performance metric only provides a partial view. For example, many probabilistic optimizers, such as genetic algorithms, vary in the quality of solution they provide due to a randomised starting point and the computation time expended by them. Accordingly, both of resulting values from the approach for the objective function and the computational time are the main performance measures in our benchmarking approach.

Providing the probabilistic nature of the optimization techniques employed by the SSSP approaches, and the modification on the algorithm they propose, precision is an important quality

indicator to be used [117] in addition to the performance measures. Precision of an approach is a subject for investigation however, it cannot be investigated using a single experiment's result. Preciseness requires a set of experimental data to measure whether they are close to each other or not [117]. Nonetheless, it is important to determine how much data is enough for the measures to provide significant results [109]. For that reason, the amount of results that each approach should provide for the comparison has been previously addressed in Section 3.2, which was illustrated to be from 30 to 100. Accordingly, results over multiple runs are required in order to measure the SSSP approaches' preciseness and stability. Standard deviation is proposed for that purpose having data of multiple runs for both estimated project time and computational time.

Accuracy of a SSSP approach on the other hand, providing the results of multiple runs, should be determined. Many studies have been concerned about the SSSP problem and SBSE in general such as [17] and have discussed whether the accuracy should be an aspect for the approaches' comparison or not. Most of these studies have shown less attention to this aspect, for example in [17] the accuracy was rejected as one of the comparison aspects for the search-based approaches. This view is based on the fact that the SBSE approaches in general are made to approximate the solution for a given problem with less computation time. However as different allocation methods have been proposed by the SSSP approaches, and each approximates the solution with different modifications and settings, the comparison between these approaches can demonstrate which one of these approaches can come closer to the optimal solution. Therefore, we have provided a basic measure for the accuracy, and have incorporated the mean of error measures for this reason. To get a more complete insight into the quality and performance of SSSP approaches we propose to use the following measures:

3.6.1 Estimated Project Time (EPT). Any approach deals with optimizing the resource allocation should come up with an allocation plan that provides the optimal or near optimal value for the objective function(s). The objective in the SSSP problem is a cost function searching for the minimum estimated project time span value. This value can be recorded from the outcome of the objective (fitness) function, i.e. the identified optimal result by an approach. While multiple runs should be made on these approaches, the results of the objective function for each run of an approach should be recorded. The EPT value therefore should be automatically stored for each run of each approach.

3.6.2 Minimal EPT (Min EPT). The minimal EPT is identified as the least possible value for estimated project time EPT among the collected values over multiple runs of a single

approach. This measure should provide an insight into the approach's capability to find a solution close to the optimal one.

3.6.3 Computational Time (CT). Computation time is the time consumed by the system to perform a SSSP approach from the point of feeding the data to the time of identifying (heuristically) the optimal result. This measure is a very important indicator for how fast the approach can be in providing an optimal or near optimal solution. The value of CT for each approach can be measured by involving a loop timer to start at the same time when the targeted approach starts, and ending when the execution of the optimizer ends, and subsequently the value of this timer defined as a CT is stored in a separate file.

3.6.4 Arithmetic Mean. The arithmetic mean is the most useful measurement to capture the average of multiple observations. The arithmetic mean can be defined as the average of a set of values having their sum divided by their number. In [17], they have shown how the arithmetic mean is a fair measure for comparison to demonstrate the relative efficiency for the cost function of an approach. This measurement is used in this thesis to capture the average of EPT values for each SSSP approach over the multiple runs. The resulted Mean value for each approach supported by the results of the standard deviation described in the next Subsection 3.6.5 can demonstrate its behaviour compared to the others. The mean denoted by $\bar{\mu}$ can be measured having the number of runs denoted by X, and the result for each run i denoted by V_i using the following Equation 11.

$$\bar{\mu} = \frac{1}{x}\sum_{i=1}^{x} V_i \tag{11}$$

3.6.5 Standard Deviation (STDEV). Standard deviation is a measurement that calculates the amount of variation between a set of observation values. The aim of using this measurement in this thesis is to demonstrate the behaviour of SSSP approaches by capturing the amount of deviation from the average and the variation among the collected EPT values that each approach produces. STDEV can serve as a predictive indicator by providing whether the potential results of an approach might be closely grouped to each other, or not. This measurement therefore is a useful metric indicator of the approaches precision and stability. Both stability and precision metrics can be measured using the same STDEV equation, yet each has its own interpretation for the outcome quality. STDEV measurement denoted by σ requires several inputs for its equation. These inputs are the number of runs denoted by X, the result for each run i denoted by V_i, and the outcome of the mean denoted by $\bar{\mu}$ from the previous equation

1. Having all the values for these inputs the STDEV can accordingly be measured using the following Equation 12.

$$\sigma = \sqrt{\frac{1}{x}\sum_{i=1}^{x}(V_i - \bar{\mu})^2} \tag{12}$$

3.6.6 Mean Arctangent Absolute Percentage Error (MAAPE). This measure, proposed by [112], is an improvement over the MAPE accuracy measure using the arctangent (inverse tangent) function. Both measures can be used to forecast the accuracy of a model, process, approach, etc. The percentage of error using MAPE measure can be calculated as follow. For x experimental runs, the obtained fitness value denoted by f should be recorded for each. In addition, the optimal value denoted by p for the experiment complexity level should be defined. Using these variables, the MAPE accordingly can be measured for a complexity level using the following Equation 13.

$$\text{MAPE} = \frac{1}{x}\sum_{r=1}^{x}\left|\frac{p_r - f_r}{p_r}\right| \tag{13}$$

MAPE however has limitations that can be illustrated by the following two situations. The first one is when the actual values are close to zero, then the outcome can go to infinity. The second situation is when the fitness values are higher (Overestimated) than the actual one, then it will result in a negative outcome value. MAAPE has overcome these weaknesses and accordingly, we have used MAAPE measurement to demonstrate the accuracy and effectivity of each SSSP approach having the arctangent function bounded the range to overcome the limitation of MAPE. Note that the arctangent or inverse tangent function is denoted in this measurement by "arctan". The percentage of error using this measurement can be calculated as follow. For x experimental runs, the obtained fitness value denoted by f should be recorded for each. In addition, the optimal value denoted by p for the corresponding experiment complexity level should be defined from Section 3.4.2. Using these variables, the MAAPE accordingly can be measured for a complexity level using the following Equation 14.

$$\text{MAAPE} = \frac{1}{x}\sum_{r=1}^{x}\arctan\left(\left|\frac{p_r - f_r}{p_r}\right|\right) \tag{14}$$

3.6.7 Optimality of Solution (Accuracy): This measurement is developed to capture the quality of SSSP approaches. The quality metric subject for exploration is the accuracy of the approaches. Accuracy should be measured based on the EPT results for 100 runs of

an approach and the optimal value for the corresponding complexity level defined in Section 3.4.2. Based on these variables the optimality of an approach can be calculated using the MAAPE to forecast the error. The forecasted error by MAAPE can lead then to forecast the accuracy level of a SSSP approach by converting the error percentage into the area of accuracy. The accuracy can be measured by the following Equation 15.

$$C = (1 - MAAPE) * 100 \tag{15}$$

From Equation 15, accuracy C is equal the subtraction of MAAPE value from one, and multiplied by 100. For instance, if the error forecasted for an approach to particular complexity level is 0.15, the accuracy of this approach is $C = (1 - 0.15) * 100 = 0.85 * 100 = 85\%$.

3.6.8 Computation Time (CT) Score: This measurement is developed to evaluate the performance of SSSP approaches in terms of computation time. This measurement is a score model that demonstrate the relevance of the computation time for each approach corresponding to a particular SSSP class by capturing the proportion of the computation time for an approach V_{ct} to the slowest among all known SSSP approaches capable of solving this class depicted by Max_{Class}. The computational time performance of an approach can be calculated then using the following Equation 16.

$$\text{CT Score} = \left| \frac{\text{V}_{\text{ct}}}{\text{Max}_{\text{Class}}} - 1 \right| * 100 \tag{16}$$

In Equation 16, the absolute value of subtracting the proportion of the computation expended by an approach V to solve a SSSP problem complexity represented by V_{ct} under consideration of Max_{Class} from one are used to measure CT score represented by a percentage value by multiplying it by 100. This measure of CT score provides better indicators of the approach's performance for the comparison analysis by a clear value that demonstrate the percentage of the performance for each approach.

3.6.9 Hit Rate: Hit rate is the capability percentage of a model, or approach to return a feasible solutions. The use of this measure is motivated by the work of [15]. The work in [15] have used it to show evidence of their approach's solution effectiveness against the work presented in [14]. This measure is adopted in this thesis to demonstrate the performance of SSSP approaches in finding feasible solutions among multiple runs. For R number of experiment runs, the hit rate for an approach having the value for each runs

r the value 1 if the outcome in that run is feasible solution or 0 if not denoted by f_r, can be then calculated using the following Equation 17.

$$Hit\ Rate = \frac{\sum_{r=1}^{R} f_r}{R} * 100 \quad (17)$$

Using the above Equation 17, if we have 30 runs and 6 out of these runs had feasible solutions, then the hit rate for this experiment is equal $\frac{6}{30} * 100 = 20\%$.

3.6. Summary

This chapter has introduced the aspects that should be considered while performing benchmarking and comparison of SSSP approaches. The main aim for benchmarking is to provide a baseline for valid experiments in software engineering research, facilitate comparative evaluation of research approaches, and to be generalized for wider research areas [114]. Therefore, the benchmarking approach presented in this chapter involve procedures and process to be followed, classification of the approaches, datasets with complexity levels of attributes and optimized (optimal) solution for each, and quality and comparison measurements for benchmark. In addition, this chapter has shown the application of mixed-methods approach, which has emerged in the last decade to the best of exploring and investigating software engineering studies. The use of this approach has improved the development of the research process carried out for this thesis.

To mitigate bias and make convincing argument a high degree of validation should exist [114]. Therefore, software engineering research should be supported by external validation capable of highlighting the application issues of the benchmark to other scenarios, so enough evidence to support claims of outcomes and generality of use can be established [114]. The main issue in this benchmark approach is the lack of coverage of software project and resource attributes. This issue can be summarized by the development activity involved within the tasks such as designing, testing, etc. and their corresponding capability by resources of personality and team factors for team formation such as Belbin factors as in [62]. In addition, as the main stream of SSSP approaches include cost of software project to the optimization problem in addition to time span, salaries of project resources therefore, is another issue that should be combined within the benchmark.

Moreover, classification of the approaches is developed on the selected attributes of software project so the more attribute the dataset has, will lead to more complexity of inputs and problem formalization levels to classify. Furthermore, qualitative analysis and statistical methods might be

applicable especially in the case of comparing solutions' quality of the actual resource allocation such the schedule organization, and utilization of resources.

It is noteworthy that software engineering benchmarking approaches as argued by [104] should comply with seven factors, which are accessibility, affordability, clarity, relevance, solvability, portability, and scalability. The benchmark proposed in this chapter complies with these factors as follow. Accessibility of the benchmark approach and all its parts including the datasets are made available throughout the thesis chapters, so the reader can easily adopt it. As this research field has not yet reached the maturity level where the approaches can be developed in tooling and technology that users can benefit their use, affordability might not be applicable as the cost of performing the benchmark associated with time consumption and performance of the approaches for multiple runs is high. The core of the benchmark process is to capture the approaches' performance where the estimated time of software projects is the main concern. Accordingly, the datasets' relevance is depicted by the different circumstances that software projects are limited to and can combine correlated information that are provided by the classification of SSSP. Datasets solvability factor on the other hand, has been demonstrated by the project information simplicity that make the comparison achievable and able to demonstrate the capabilities of various SSSP approaches too. Moreover, portability can be achieved by the capacity of the benchmark process to hold additional optimization objectives as well as software project and resource's attributes that can be combined within the datasets with evolving classification for new attributes to scale up for different maturity levels of software project circumstances.

Our benchmarking approach has been applied on a set of SSSP approaches, where the details of these approaches, and the outcomes of using the proposed quality measures and the datasets levels are detailed in the next Chapter 4.

Chapter 4 Evaluation of Nine SSSP Approaches

This chapter evaluates a set of SSSP approaches by applying the benchmarking approach from Chapter 3, and presents the experiment's result, and findings.

4.1 Introduction

Since late 90's, different approaches have been proposed to solve SSSP problem such as [14, 18, 22]. These approaches however are more formally designed to explore the optimization techniques, their potential capability, strength, capacity, and how they can be used in approximating and solving software project management problems considering different optimization objectives. Benchmarking and evaluating these approaches has become more important to present their capability for next generation research and to provide future direction on potential points of interest for consideration in minimizing software project completion time. Therefore, this chapter adopts the benchmarking process presented in Chapter 3 to benchmark and evaluate a set of SSSP approaches. These approaches are selected based on a four points criteria discussed later in the following Section 2.4.5.

The reminder of this chapter is divided into three sections. The following Section 2.4 provides description and background of different SSSP approaches. The study aims and research questions are listed in Section 4.2. In Section 4.3, results for each experiment on each selected SSSP approach and its outcomes analysis regarding efficiency, effectivity, performance, and accuracy are presented. Throughout this section it will be much clearer to the reader why the benchmark dataset is divided into different levels and how they are connected with the constraints to provide a taxonomy for the SSSP problem. The conclusion with the main findings and limitations of the experiments performed for this chapter are provided in Section 4.5.

4.2 Experiment Aims and Parameters Settings

Throughout the development of the benchmarking approach and preparation for this chapter, question were raised about the validity of the benchmarking approach and the performance of SSSP approaches. The experiments performed for this chapter constitutes the answer for these questions, which are the following:

1. Do the SSSP approaches perform similarly?
2. If no. What are the differences between the approaches? and
3. Do the measures adopted for the benchmarking able to demonstrate the approaches' performance and quality?
4. Does the classification made for the approaches able to demonstrate the performance, capacity and capability of the approaches as the complexity increases?

The experiments provided in this chapter are performed according to the benchmarking approaches described in Chapter 3 using Intel Core2 Quadm5 (2.66 Ghz) CPU, supported by 4GB memory. The implementation of the approaches selected for this study is carried out using Matlab 2013a, supported by global optimization toolbox to facilitate the development of the optimization techniques proposed by these approaches. Each approach was executed 100 times according to the benchmarking approach to allow determination of mean and deviation values. The experiment settings for each approach and complexity level are defined according to the description of each from the last section. The parameter settings of each selected approach for the experiments are presented in the following Table 19.

Table 19: Parameter Settings of the Selected Nine SSSP Approaches

Approach	Settings				
	Population size	**Generation**	**Crossover fraction**	**Mutation probability**	
Chang01	60	500	0.65	0.15	
Antoniol01	100	1000	0.6	0.1	
Antoniol02	100	400	0.6	0.1	
Alba01	64	5000	0.9	0.01	
Ren01	50	100	0.6	0.2	
DiPenta01	50	250	0.7	0.1	
Minku01	64	79	0.75	0.01	
Park01	100	400	1	0.05	
	Initial Temp	**Control**	**Cooling**	**loops**	
				Internal	**External**
Kang01	100	2000	0.95	500	8

The parameter settings presented in Table 19 above are motivated by the best experiment settings used for the approach's performance and outcomes validation provided by the authors of these approaches.

4.3 Results

This section provides detailed results of each approach selected for this study according to the corresponding complexity level that the approach can perform. Noteworthy that due to the approaches capability for solving different complex scenarios the approaches presented will be gradually reduced as the complexity increases. This section will be grouped according to the complexity and classes of SSSP. It is important to notice that two things are to capture in this section which are:

- How each approach performs compared to the rest in the same complexity level? and
- How each approach performs as the level of complexity increases?

An observation was made during the experimentation of the selected approaches that some of them have performed badly in terms of returning feasible solutions. When there is no feasible result recorded for an approach, then this value is recorded as "NA". This result can be acceptable as the constraints in some approaches have been set to a certain value to make sure that the approach can be able to return a feasible solution. Therefore, we use the hit rate (number of feasible solutions) for those approaches that might return unfeasible solutions, so their accuracy can be demonstrated.

4.3.1 Complexity Level One Experiments

Nine SSSP approaches are subjects in this complexity level to test their performance and accuracy outcomes. As this level corresponds to the first class of SSSP problem, the only information that it provides for the SSSP approaches to search for an optimal or near optimal time estimate is the estimated effort and the number of resources available to the software project. The dataset of this complexity level is provided by Section 3.4.2. The following Figure 17 depicts the results of EPT values for each approach in this level using the Boxplot diagram.

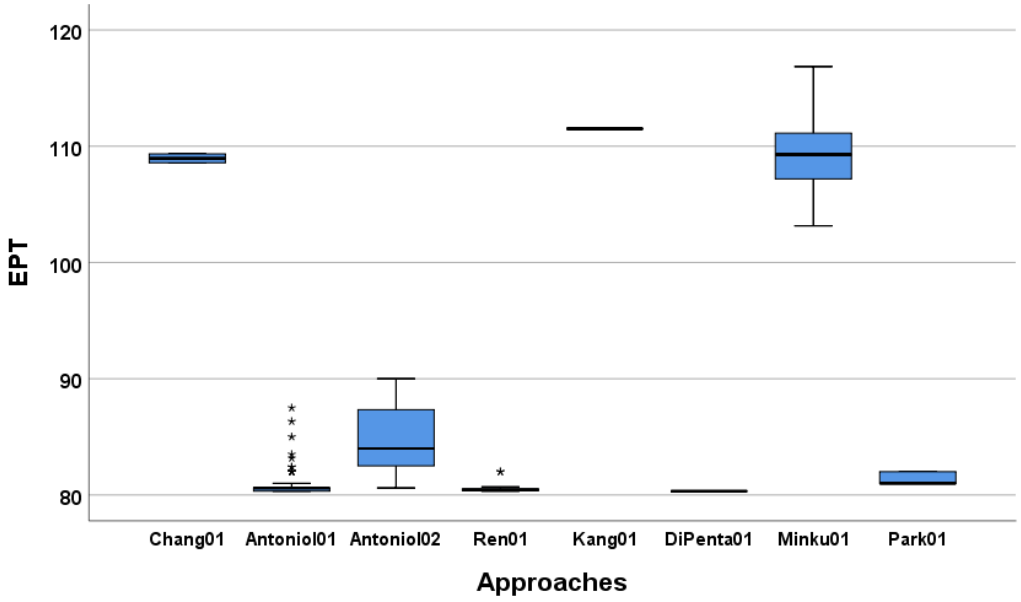

Figure 17: Level One Boxplot Diagram of SSSP Approaches Evaluation

It can be seen in Figure 17 that three approaches of (Chang01, Kang01, and Minku01) are the worst among the others. However, it should also be seen that the approach of Alba01 is missing in this figure as it failed to provide any results over the 100 runs. On the other hand, five approaches of (Antoniol01, Antoniol02, Ren01, DiPenta01, and Park01) by this figure can be seen as better than the earlier mentioned approaches, providing estimates between 80-90 days. In addition, it can be seen in this figure too, that DiPenta01 is able to provide precise, and less variation EPT results than any other approach.

Table 20 provides more detailed information about the approaches' outcomes, which can enable us to see which of these approaches can outperform the others. The approaches' in this table are sorted by the EPT values, from the worst to the best.

Table 20: SSSP approaches Results for Complexity Level One

Approach	EPT	CT	Hit Rate	*CT Score*	MAAPE	Accuracy
Alba01	NA	220.14	0	96.35	NA	NA
Kang01	111.5	127.91	100	97.88	0.370	62.99
Minku01	109.19	10.74	100	99.82	0.345	65.54
Chang01	108.95	18.40	2	99.7	0.342	65.77
Antoniol02	85.13	109.66	100	98.18	0.060	94.04
Park01	81.31	6033.43	100	0	0.012	98.78
Antoniol01	80.83	285.92	100	95.26	0.006	99.37
Ren01	80.48	17.57	100	99.71	0.002	99.81
DiPenta01	80.33	24.69	100	99.59	0.005	99.99

Table 20 presents the mean of Estimated Project Time (EPT) and Computation Time (CT) outcomes over 100 runs of each approach. In addition, it provides information regarding hit rate, MAAPE, and accuracy for each approach. The unit of EPT value is in days, and for CT is in seconds. For example, Minku01 approach has provided 109.19 days average EPT value for 100 runs. This approach can provide feasible solutions with Hit Rate of 100 times out of 100 runs. In addition, this approach consumed 10.47 seconds on average, and scored 99.82% of the average CT compared with the worst approach performance among them all. The error forecasting of this approach using MAAPE measure shows that around 34.5% of the approach's outcome is prone to overestimate EPT, which leaves the approach with 65.5% accuracy. The worst performance among them all is the Alba01 approach. This approach has failed to provide a single estimate over the 100 runs and consumed on average 220 seconds on searching for a solution, which makes it the worst approach of this complexity level.

Moreover, the approach in Chang01 has a very high score of CT, however, that comes at the cost of accuracy of EPT. On the other hand, the one in Park01 with the worst CT was able to provide a more accurate EPT. This shows how reducing computation time can come at the cost of good quality solutions in Meta-Heuristics. However, this fact can no longer be valid as with DiPenta01 approach, which has shown its capability to outperform the others according to both CT score and EPT Accuracy. This approach has shown its stability of producing precise results over the 100 runs, and so far, that make it the one that outperform the others in this particular complexity level.

To capture whether these approaches perform similarly, we have performed a paired T-Test against the one of DiPenat01. Our null hypothesis is that the approaches can provide similar estimates and perform similarly. The results of this test are depicted in the following Table 21.

Table 21: Level One Paired T-Test of SSSP Approaches Evaluation

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	Dipenta01 - Chang01	-28.61905	0.53875	0.38095	-33.45950	-23.77859	-75.125	1	0.008
Pair 2	Dipenta01 - Antoniol01	-0.49983	1.13947	0.11395	-0.72592	-0.27373	-4.386	99	0.000
Pair 3	Dipenta01 - Antoniol02	-4.79867	2.60522	0.26052	-5.31560	-4.28173	-18.419	99	0.000
Pair 5	Dipenta01 - Ren01	-0.14594	0.24051	0.02405	-0.19366	-0.09822	-6.068	99	0.000
Pair 7	Dipenta01 - Minku01	-28.85713	2.51428	0.25143	-29.35602	-28.35824	-114.773	99	0.000
Pair 8	Dipenta01 - Park01	-0.97667	0.46482	0.04648	-1.06890	-0.88444	-21.012	99	0.000

From Table 21, it can be seen that the difference in mean for each pair of DiPenta01 approach against the others has a 2-tailed value less than 0.001, and for the first pair the significance was

with 0.008. From these results we have found enough evidence to suggest that the difference between the two scores for each pair is statistically significant and reject the null hypothesis.

4.3.2 Complexity Level Two

This level provides information about the estimated effort and precedence relationship between the project tasks, as well as the number of resources available to the software project. The project time minimization problem is depicted by the dataset in Section 3.4.2. In this level, only six approaches are subjects to test their performance and accuracy outcomes. The reason of taking out the approaches of (Antoniol01, Antoniol02, and Kang01) is that they do not support the information provided by this level of task dependencies, and any other attributes for higher complexity levels. The following Figure 18 depicts the results of EPT values for each approach in this level using the Boxplot diagram.

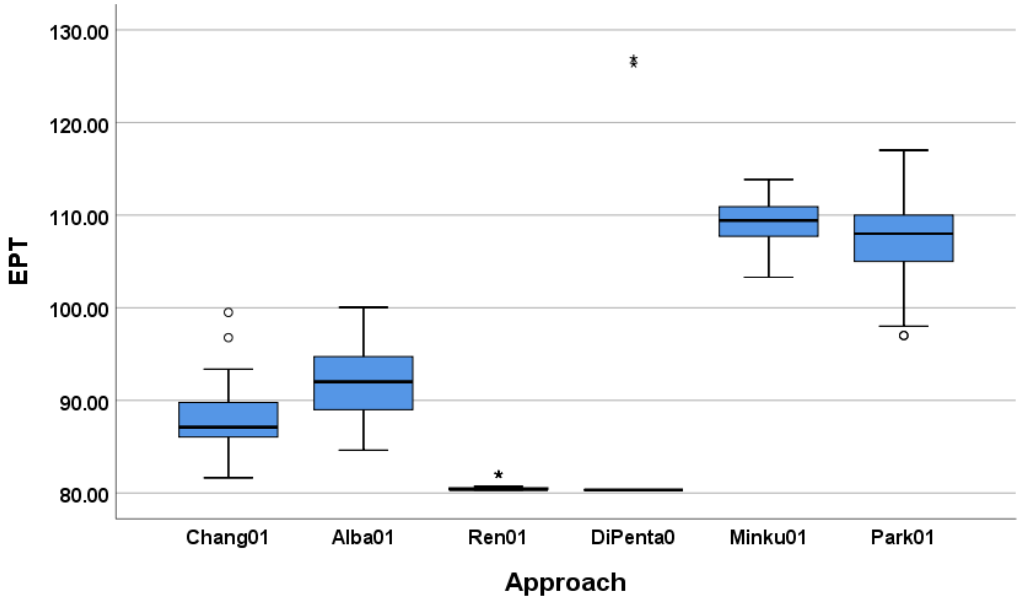

Figure 18: Level Two Boxplot Diagram of SSSP Approaches Evaluation

It can be seen in Figure 18 that two approaches of (Minku01, and Park01) are the worst among the others, with average close to 110 Days. On the other hand, the approaches of (Chang01, and Alba01) have provided better estimates than the earlier mentioned approaches, between 80-100 Days. Moreover, the approaches of (Ren01, and DiPenta01) can be seen as the best among all the other approaches according to the average of EPT with just over 80 Days, and their stable and precise EPT values. However, DiPenta01 approach had some very extreme overestimates of EPT, which make Ren01 approach, with the least variation of EPT results, outperforms any other approach on this particular level. To make this claim, we need more information regarding the CT

and other accuracy measures. Therefore, the following Table 22 presents the performance and accuracy outcomes of the approaches.

Table 22: SSSP approaches Results for Complexity Level Two

Approach	EPT	CT	Hit Rate	CT Score	MAAPE	Accuracy
Minku01	109.17	10.34	100	99.1	0.344	65.56
Park01	107.58	1133.77	100	0	0.326	67.37
Alba01	92.04	270.28	100	76.16	0.144	85.55
Chang01	87.63	21.62	100	98.1	0.090	90.95
DiPenta01	81.26	29.24	100	97.42	0.011	98.95
Ren01	80.51	25.14	100	97.78	0.002	99.77

From Table 22, it can be seen that Alba01 approach, which have struggled to provide feasible solution for the previous complexity level, is now capable to provide feasible results with 100% hit rate performance. However, this level provides a challenge for the approaches especially in terms of stability of results over the runs. The resulted value of Minku01 approach in terms of EPT is the worst amongst the approaches. This is due to the CT spent on searching the solution space, which for this approach is the least one among the rest recorded by the CT Score with 99.1%. By this speedy search, the accuracy of this approach is the worst recorded with approximately 65.5%.

Again both (DiPenta01, and Ren01) approaches in this level could be seen having the same results as seen by the previous Boxplot diagram. Yet, the mean EPT provides clearer picture of which can provide better results over the runs. In this case, Ren01 obviously performs better as it provides the average of EPT equal to 80.5 days. Consequently, the accuracy for this approach in this level is the best among the rest with score of 99.7%.

To capture whether the approaches in this level perform similarly, a paired T-Test is performed. However, this time the pairs are made against Ren01 approach, as this approach has managed to provide better estimates than the other ones. The outcomes of this test are recorded in the following Table 23.

Table 23: Level Two Paired T-Test of SSSP Approaches Evaluation

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	Ren01 - Chang01	-7.11916	3.18412	0.31841	-7.75096	-6.48737	-22.358	99	0.000
Pair 2	Ren01 - Alba01	-11.53361	3.87057	0.38706	-12.30162	-10.76561	-29.798	99	0.000
Pair 3	Ren01 - DiPenta01	-0.74820	6.54196	0.65420	-2.04626	0.54987	-1.144	99	0.256
Pair 4	Ren01 - Minku01	-28.65805	2.56428	0.25643	-29.16685	-28.14924	-111.759	99	0.000
Pair 5	Ren01 - Park01	-27.06939	4.08808	0.40881	-27.88055	-26.25822	-66.215	99	0.000

What can be seen by Table 23 is that with enough evidence to suggest that the difference between the two scores of each pairs are statistically significant and reject the null hypothesis with 2-tailed values less than 0.001, except the difference between Ren01 and DiPenta01 approaches, which do not provide enough evidence to reject the null hypothesis for them particularly. Accordingly, both approach can be seen of a similar performance.

4.3.3 Complexity Level Three

As this level provides additional project attributes that some of the approaches do not include within their problem formalization, the one in Ren01 is accordingly excluded in addition to (Antoniol01, Antoniol02, and Kang01) from this level. Therefore, only five approaches are subjects in this level to test their performance and accuracy outcomes. The dataset used for this level and its optimal solution can be found in Section 3.4.2. This dataset includes the information about the estimated effort and precedence relationship between the project tasks. In addition, this dataset includes the number and skills of resources as part of the software project data for SSSP approaches to solve its time estimation problem. The following Figure 19 depicts the results of EPT values for each approach in this level using the Boxplot diagram.

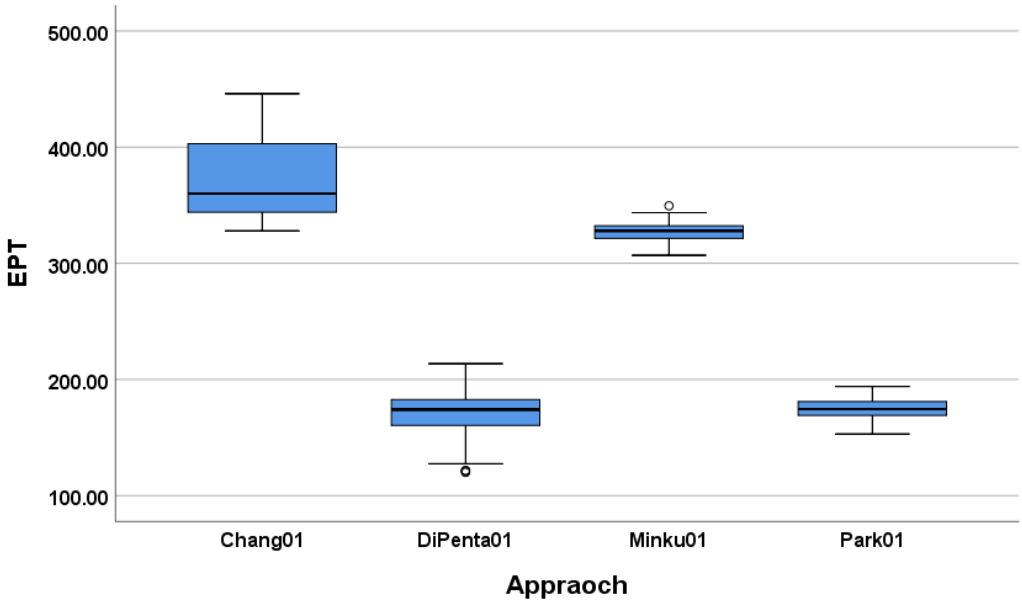

Figure 19: Level Three Boxplot Diagram of SSSP Approaches Evaluation

It is important to notice that again the Alba01 approach is not included in Figure 19, as it has again failed to provide a single estimate. From Figure 19, it can be seen that two approaches are outperforming the rest, and Chang01 approach is the worst on providing EPT values with just over 350 Days. However, in order to make clear evidence of which is good, and which is worst. A full

information regarding the performance and accuracy outcomes are required. Therefore, the following Table 24 presents the results of the approaches for this complexity level.

Table 24: SSSP approaches Results for Complexity Level Three

Approach	*EPT*	*CT*	*Hit Rate*	*CT Score*	*MAAPE*	*Accuracy*
Alba01	NA	206.62	0	98.5	NA	NA
Chang01	378.00	15.39	3	99.89	1.199	-19.89
Minku01	327.64	16.89	100	99.88	1.135	-13.51
Park01	175.25	13747.62	100	0	0.598	40.19
DiPenta01	172.75	15.60	100	99.89	0.571	42.95

It can be seen from Table 24 that both Alba01 and Chang01 are struggling in this level to provide feasible solutions. For instance, Alba01 approach is the worst amongst the approaches as it did not find any feasible solution over the 100 runs, where the one of Minku01 that is similar to those approaches have succeeded to find a feasible solution in each run exposed by 100 for the hit rate. However, the mean of EPT value for Minku01 approach is far from the optimal solution of EPT. Consequently, the accuracy of this approach and Chang01 too are considerably inaccurate to solve a problem where skills and effort of tasks are the only inputs. Yet, DiPenta01 approach still among the best ones to provide better solutions than the others for this level too. This is depicted by the accuracy and CT score values, which are 42.9% and 99.89% respectively.

Furthermore, the CT score of Minku01, Chang01, and DiPenta01 approaches have almost no difference from one to another. The one that comes slightly less with 98.5% CT score is Alba01. The CT score computed for these approaches is compared with the worst CT ever recorded among them all, which is Park01. However, this approach provided good quality EPT for 175.25 days, reflected on the accuracy result with 40.19%. Despite the fact that all the approaches in this level did not provide good quality outcomes, Park01 approach in this level can be considered as the second place for good quality results amongst the others. It can be concluded that this level has challenged the approaches on their capacity to handle skills without dependent tasks, so the allocation can be harder for those approaches that consider individuals for task assignment, leaving the best to be by DiPenta01.

But now the question is: are these approaches perform similarly? To answer this question, we have to look at the paired T-Test results that compare the approaches against DiPenat01 one for this level. These results are depicted in the following Table 25.

Table 25: Level Three Paired T-Test of SSSP Approaches Evaluation

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	DiPenta01 - Chang01	-209.98535	54.93783	31.71837	-346.45848	-73.51222	-6.620	2	0.022
Pair 3	DiPenta01 - Minku01	-154.89362	23.65115	2.36511	-159.58652	-150.20072	-65.491	99	0.000
Pair 4	DiPenta01 - Park01	-2.50438	22.89521	2.28952	-7.04729	2.03853	-1.094	99	0.277

What can be seen by Table 25 is that for both Chang01 and Minku01 approaches againstDiPenta01 there are enough evidence to suggest that the difference between the two scores of each pairs are statistically significant and reject the null hypothesis with 2-tailed values less than 0.001 for Pair3 of (DiPenta01 and Chang01), and 0.05 for Pair1 of (DiPenta01 and Minku01). It is noteworthy that Alba01 approach is not included in this test, as it could not provide solution for this level. On the other hand, the difference between DiPenta01 and Park01 approaches for this level do not provide enough evidence to reject the null hypothesis. Accordingly, both approach can be seen of a similar performance.

4.3.4 Complexity Level Four

Like the previous complexity level, five SSSP approaches are subjects to test their performance and accuracy outcomes for this level too. The data for this level provides information about the estimated effort and precedence relationship between the project tasks. In addition, it provides information about the number and skills of resources available to the software project in order for the SSSP approaches to search for an optimal or near optimal solution of time minimization. The dataset used for this level can be found in Section 3.4.2. The following Figure 20 depicts the results of EPT values for each approach in this level using the Boxplot diagram.

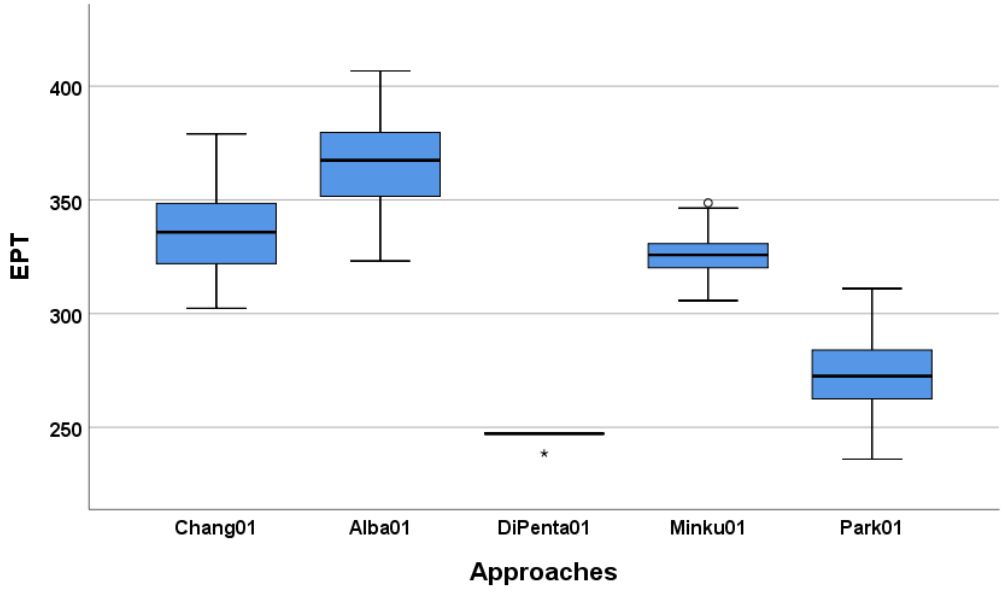

Figure 20: Level Four Boxplot Diagram of SSSP Approaches Evaluation

It can be seen in Figure 20 that Alba01 approach has successfully provided EPT solutions for this level. However, the mean EPT for these solutions is much higher than any other approach. Accordingly, we can say based on this figure that Alba01 approach is the worst among the approaches in this level too. On the other hand, DiPenta01 approach has almost no variances in its estimates, and these estimates are the least amongst all the other approaches. Therefore, DiPenta01 approach can be counted as the best approach for this level too. However, to make this claim, we need more information regarding the CT and other accuracy measures. Therefore, the following Table 26 presents the performance and accuracy outcomes of the approaches.

Table 26: SSSP approaches Results for Complexity Level Four

Approach	*EPT*	*CT*	*Hit Rate*	*CT Score*	*MAAPE*	*Accuracy*
Alba01	365.25	253.26	100	92.76	0.665	33.53
Chang01	338.04	20.05	100	99.43	0.577	42.29
Minku01	326.34	10.51	100	99.7	0.538	46.21
Park01	272.81	3499.41	100	0	0.322	67.82
DiPenta01	247.09	23.87	100	99.32	0.206	79.36

In this level, it can be seen from the values presented in Table 26, how the approaches perform when dealing with multiple problem factors that adds up dependency constraint and skills of resources together, so the feasible area within the search is limited. This level is clearly creating more complexity for the approaches than in the previous level to provide accurate solution, in which their accuracy do not provide better or even similar degree as to level one and two.

As dependency is one of the problem factors in this level, the approaches that have performed badly in the previous level are now providing 100% for hit rate, such as Alba01. However, their solution quality is varied and can be captured from their accuracy and CT score. The one in Alba01 for instance has 33.5% accuracy and 92.7% CT score, where the one in Chang01 has made better progress in accuracy with around 42.3% and for CT score with 99.4% too. Accordingly, Alba01 can be counted as the worst amongst all the other approaches.

Furthermore, the one in Park01 has again provided a reasonable degree of accuracy and stabilized solutions over the runs compared with the others, however, it has performed badly in terms of CT. This performance of Park01 has made the other approaches to have much higher CT score.

It is important to record that the outcomes of Minku01 approach demonstrate how it has made good improvements over the work of Alba01 and Chang01 too in two aspects. The first one is the accuracy, which recorded for Minku01 with 46.2%. This is clearly not a convincing accuracy but compared with both Alba01 and Chang01 results, it shows an evidence of Minku01 outperforming them. The second aspect is the time, which for Minku01 has a time cost for each run around 10.5 seconds. This value is half the time spent by Chang01, and far less from Alba01 that spent around 253.3 seconds.

With slightly less CT score than Minku01, DiPenta01 has managed to provide with 79.35% accuracy a better solution of around 247.1 days for EPT than all the other approaches, and it can be concluded that DiPenta01 outperform all the other approaches of this level of complexity.

But now the question is: are these approaches perform similarly? To answer this question, we have to look at the paired T-Test results that compare the approaches against DiPenat01 one for this level. These results are depicted in the following Table 27.

Table 27: Level Four Paired T-Test of SSSP Approaches Evaluation

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	DiPenta01 - Chang01	-90.94956	18.33695	1.83369	-94.58801	-87.31111	-49.599	99	0.000
Pair 2	DiPenta01 - Alba01	-118.15420	18.58804	1.85880	-121.84247	-114.46593	-63.565	99	0.000
Pair 3	DiPenta01 - Minku01	-79.24830	7.77128	0.77713	-80.79029	-77.70631	-101.976	99	0.000
Pair 4	DiPenta01 - Park01	-25.71695	16.45139	1.64514	-28.98126	-22.45264	-15.632	99	0.000

From Table 27, it can be seen that the difference in mean for each pair of DiPenta01 approach against the other has a 2-tailed value less than 0.001. Therefore, we can conclude that there is enough evidence to suggest that the difference between the two scores for each pair is statistically

significant and reject the null hypothesis of having similarity of performance between the approaches.

The overall findings from the performance and accuracy outcomes of each approach for all the complexity levels have shown that some of the approaches performed badly, others were moderate, and limited approaches were capable of providing good quality solutions of project time estimation. The overall weaknesses and strength of each approach that encountered by the outcomes of all the levels, and highlighting the best are presented in the following Table 28.

Table 28: Overall Findings from the Complexity Levels for each SSSP Approach

Approach	Encountered Weaknesses or Strengths
Chang01	This approach can only perform better when dependency is existing between project tasks.
Antoniol01	It can provide estimates for only level one.
Antoniol02	It can provide estimates for only level one.
Alba01	This approach can only perform better when dependency is existing between project tasks.
Ren01	This approach can provide a very good estimate however, it can only work for complexity level one and two, and provide the best for level two.
Kang01	With its overcomplicated settings, this approach provides a moderate estimate, and only for level one.
DiPenta01	**This approach has dominated the four complexity levels with the best results.**
Minku01	This approach can only perform better when dependency is existing between project tasks.
Park01	This approach consumes computation time of over 18 minutes to provide a single estimate.

What it can be concluded is that some approaches were capable on providing solution only for limited complexity level(s), such as both Antoniol01, 02, and Kang01. Others, such as Alba01 and Chang01, have failed to provide solutions for simpler levels, and for higher complexity levels their performance was much better. What is noteworthy is that DiPenta01 was the only approach that is capable of providing good quality solutions over the complexity levels, which make it the one that dominate the others.

4.4 Analysis

Three approaches of (Chang01, Alba01, and Minku01) are of a major concern with their solution quality. The one of Alba01, for example, has failed on every run to provide a single solution for complexity level one and three, and had the worst performance among the approaches of level four. This approach has several issues that have contributed to this bad performance. The first one is the allocation method that it uses to allocate the resources. This method assigns resources to tasks with a participation percentage. That means the resource will work on the task for a percentage of his/her day time. This will be reflected on the time estimate of that task, where the

effort won't be divided by the number of resources, but on the overall percentage of the assigned resources to that task.

Moreover, in the approach's settings identification presented in [14], the resource can be allowed for overtime work limit with an overall participation of 120%. However, in our experiments we did not allow this value and assumed that the resource can only work with a full of 100%. In addition, the creation of solutions in Alba01 is structured by a 2-D matrix, where the rows are representing the resources and the columns are representing the tasks. The values associated to the cells are randomly generated between {0, 1} representing the amount of the participation percentage of the resource to that task. Based on this representation, each resource by this matrix will have a percentage to participate with for each task. Having no dependencies between the project tasks by the datasets representing complexity levels one and three, means that the tasks will be performed at the same time. Therefore, the solution, in general, will include some values that exceeds the maximum participation of 100% and violate the overtime constraint. Thus, making the solution unfeasible, as it is hard to find participation percentages that can works for all the resources to be assigned to project tasks without causing them to work overtime.

On the other hand, Kang01 has provided the worst estimates on level one, and failed to compete for the other levels. The reason for this approach to fail on continuing the higher levels is that it does not support dependency and technical software development skills aspects such programming languages. Moreover, this approach allocates the resources with an initial plan generated by a greedy algorithm. This algorithm starts by sorting the tasks from larger to smaller size, and continuously assigning a resource to each task, and move to the other, till all the resources are assigned to tasks and all the tasks have been allocated with at least to one resource. What left then for the simulated annealing algorithm to do in this approach is to re-assign the resources with percentages, and to the most fitted task. By this type of assignment, the best plan for the first level is to keep the same amount of resources assigned without reducing their participation percentage. Therefore, this approach has assigned two resources to task 2, 3, 4, and 5, and a single resource to each of the rest. While project time for this complexity level can be counted as the maximum task length among the others, task two with estimated effort of 223 has the maximum time of 111.5 Days among the others, which is the estimate value provided by this approach. This approach has shown how using a heuristic technique for initiating a population can fail the algorithm by having similar solutions and little diversity.

Minku01 approach, moreover, had the worst outcomes for level two. Broadly speaking, one of the reasons for (Chang01, Alba01, and Minku01) approaches to perform badly is the absence of dependencies between the project tasks as they perform the allocation with assigning participation

percentage to resources. However, this time the dataset of level two holds dependency information about the project tasks. What makes this approach unable to compete with the other approaches are by two main reason. As with the Alba01 approach, Minku01 assigns the resources with a participation percentage. Moreover, the total project time in this approach is computed by cumulating the overall tasks' time. Accordingly, Minku01 provides a project time estimate where all the tasks are counted, and not the longest path among the paths of the TPG. For example, one of Minku01 solutions is depicted by the following Table 29.

Table 29: Minku01 Allocation Example

	T1	T2	T3	T4	T5	T6	T7	T8
R1	0.000	0.857	0.857	1.000	0.714	0.286	0.857	0.429
R2	0.571	0.857	0.857	0.857	0.857	1.000	0.429	0.857
R3	0.714	0.857	1.000	0.571	0.714	1.000	0.714	0.571
R4	1.000	0.857	1.000	0.429	0.714	0.857	0.714	0.571
R5	1.000	0.857	0.857	0.714	0.571	0.857	0.286	0.857
R6	0.857	1.000	0.429	1.000	0.571	0.429	0.000	0.286
R7	0.571	0.571	0.857	0.571	0.714	0.857	0.571	0.143
R8	1.000	0.857	1.000	0.857	0.857	1.000	0.571	0.571
R9	0.429	0.857	1.000	0.714	1.000	0.857	0.714	0.286
R10	1.000	0.857	0.286	0.714	0.857	0.714	0.571	0.571
R11	0.571	1.000	0.571	0.429	0.429	0.857	0.143	0.714
R12	0.571	1.000	0.857	0.714	0.714	0.286	0.714	0.714

According to the participation percentages that each resource will do for each task in Table 29, the estimated time for task 1, 2, 3, 4, 5, 6, 7, and 8 will be 9.89, 21.38, 18.8, 15.4, 21.8, 5.5, 9.86, and 6.84 Days. By adding all these estimates together the project time will be 109.5 Days. It is noteworthy that what makes Minku01 approach to sustain with 100% Hit Rate is that in case of a resource working on more than one task and those tasks are in parallel, the overall participation of this resource for these tasks are divided by the number of these tasks.

The results, presented in last Section 4.3, show that Park01 approach had a very long run of CT. The performance of Park01 approach, for example, has an average of 13747 Seconds for level three. The reason for this approach to perform like this is the fitness function and GA settings adopted by this approach. The fitness function simulates project time day by day. This means that a part of the fitness function is iteratively computing each task's time by the same for loop, where the computation of every edge of the TPG, parallel tasks, and waiting tasks to be performed are all in as nested loops. Therefore, this loop complexity can be described as $O(n^2)$, and this is the reason why the CT of this approach is the worst among all the other approaches.

4.5 Conclusion

The results provided in this chapter demonstrate the differences between various SSSP approaches selected for the comparison. These differences are presented in the form of performance, accuracy, and capability of the approaches for different complexity levels.

The first conclusion from the results has answered the first research question as whether the approaches perform similarly. From the results, it is clear that there are differences between the approaches selected for the comparison. It can be seen too that the differences between the approaches are demonstrated by the metrics of CT and EPT, where the measures of error, accuracy, and CT scores have exposed more about the differences in the approaches' quality and performance. Accordingly, this provides the answer for both second and third research questions. Furthermore, the results of the approaches are differ from one complexity level to another by which some approaches failed to provide a single result over multiple runs in one level, and in another level performs with 100% hit rate. According to these results, the classification made and the derived complexity levels combined with the accuracy measures are able to demonstrate the performance, capability, and capacity of SSSP approaches, which answering the fourth question.

For complexity level one, the approaches in [20, 22, 94] provide very close accuracy to each other, but they differ in terms of CT. However, the one in [22] outperform the others in terms of MAAPE and accuracy. It is also important to see how some of the approaches are unable to solve further complexity levels, and consequently they are omitted from further experiments as the complexity level increases. For complexity level two, a clearer winner can be identified with the one in [94]. However, the one in [22] has less error over multiple runs. For level three, a clearer winner can be identified with the one in [22] offering similar accuracy to the one in [23], but requiring far less time. For level four, again the one in [22] wins in terms of accuracy, and performance over 100 runs experiments according to the measures of mean of EPT, STDEV, and MAAPE for results' accuracy and stability. It can be concluded that some of the approaches specially the one in [22] outperform the others almost in every level for both CT and EPT, as well as the related measures of accuracy and CT score.

It is also worth mentioning that the approaches that do not consider skills nor productivity supposing that human resources possess the same skillset and productivity, have the matter of searching for best resource allocation, but with no difference of which resource to use while forming teams and allocating them to project tasks. The only scenario that these approaches could work for is giving an insight about the importance of scheduling when there are very limited

resources, having the same solution of COCOMO, but requiring to know how to allocate those resources to gain this time length as in [20, 21].

The main aim of SBSE as discussed by [17] is not to provide an automatic decision-making system, but to provide a tool that can support the DM's work. With one step of work towards that, this chapter has demonstrated the differences between the selected SSSP approaches for each complexity level showing which scenario and situation an approach can provides a better solution than the others. This can provide the DM with an overview of which approach can be most beneficial to which situation (s)he might face.

In addition, the results show that some of the complexity levels needs more attention on resource's skills implemented in both level three, and level four. Those two levels require a simulation of PM's choices with respect to their industrial settings as an attempt to improve the capability of the optimization techniques to explore more beneficial and feasible solutions.

As it is the cases in software engineering and many other fields, the researchers might find some concerns about the work carried out and the implementation of the work that could limit its outcomes or affect its validity. This chapter follows the benchmarking process proposed in Chapter 3, consequently the results provided are prone to the limitation discussed about this process. Despite the fact that the problem defined in this thesis is about time span minimization, software projects encompass many attributes and parameters that is required to support more real-world objectives such as resource's salary, and profession. The benchmarking approach used in this chapter is capable of supporting these attributes and objectives, however the datasets provided for the benchmark do not provide information about these attributes. Consequently this can affect the generalizability of the results and applicability to different software project problems. However, as this would reflect on the classification as mentioned in the benchmarking approach limitations, the benchmark approach is capable to adopt more classification as the problem expands.

Furthermore, effort estimation as provided in the datasets and used by the approaches is prone to errors of providing a linearity concept of time that is effort over team size as in [18, 22]. To generalize the benchmark and provide as many SSSP approaches as possible to a comparison this concern was discussed by [22] to offer generalization of SSSP approaches purpose, as many approaches have used COCOMO model [34] to provide their effort estimate as in [15, 22, 93, 118].

Additionally, one of the limitations counted in the work carried out for this chapter concerns the experimentation work of SSSP approaches as the implementation of these approaches might have implicated some changes on what they have meant to produce. For instance, dependency handling

in some approaches was unclear whether it is for one to one relationship and the precedence should be formed between one task and another, or these approaches support one to many dependencies forming a relationship between a single task with multiple ones. The later was mainly what has been implemented for all the approaches, which might not be the case at all.

One important issue is that all these approaches considered in this chapter do not support productivity as an input to SSSP problem except the one in [23]. For this reason, it is important to capture how the proposed allocation methods within these approaches will perform having resource's productivity aspect as one of the problem inputs to be solved using GA. In that sense, the next chapter provides details on the implementation and optimization problems defined for that work, as well as the outcomes and results of the allocation methods defined.

Chapter 5 SSSP with Team Formation and Distribution to Project Tasks

Following the outcomes and conclusions from Chapter 4, a comparison between the team allocation methods adopted by the set of the approaches selected to that chapter is required specially since none of these approaches have considered variability of resource's productivity as one of their problem inputs except the one in [23]. This has motivated us to demonstrate how allocation methods involving resource productivity aspect within the SSSP optimization problem parameters can be solved using GA. The aim of this work is to observe which of the allocation methods can fit with the fifth SSSP problem complexity well that presented by level five in the datasets, and the most suitable GA's settings for this problem to provide very accurate, precise, and speedy solutions.

This chapter accordingly provides to the reader a comparison between four major team allocation methods that are adopted by the SSSP approaches. These methods are combined with an optimization technique, and a fitness function to simulate project time considering dependencies between project tasks, and resources' competencies and productivity. The results of the comparison between these methods using different accuracy measures provided in the benchmarking approach -presented in Chapter 3- have shown that one of these methods outperforms the rest in two circumstances however another shows its effectiveness to handle more complicated problems.

5.1 Introduction

Project time span, cost, quality, and time to market are important measures to demonstrate whether a software development project is successful or not [8]. As one of the critical aspects in software project management, project time span can be minimized by having the suitable teams distributed to the most fitted tasks. However, finding the suitable productive and skilled resources, forming them into teams, distributing them to project tasks, and then scheduling these project tasks according to the team staffing and dependencies constraint between these tasks is a

complex problem. This problem is defined in Section 1.2 as Staffing and Scheduling a Software Project (SSSP), which has been researched by many approaches in the last three decades using different optimization techniques. This problem is well-known as an NP-Hard problem complexity, by which using an exhaustive search method to solve it requires forbiddingly long execution time as the problem size increases [18]. Accordingly, approximation of results using a meta-heuristic techniques can be acceptable as a trade-off of accuracy to near optimal solutions for less computation time.

Staffing the resources according to the project constraints requires consideration of alternative allocations of resources to tasks. Assigning resources individually in software project environment can be an option. However, teams are the essential element in software development and production [28, 119].

Based on our taxonomy for the level of complexity of the software project's information, and the results of the experiments performed against the nine SSSP approaches, there have been an urgent need to classify the main aspect that contributes to the success of a SSSP approach. What we have found that two major methods for allocating human resources to teams and tasks in software projects are dominating the approaches that optimize the SSSP problem. These methods are the most distinguished feature between the SSSP approaches. Based on this finding, we have created four team allocation methods, where three of them are comply with those used by the approaches, and a new allocation pattern based on our understanding to those methods used. These allocation methods are the first contribution of this chapter.

On the other hand, formalization of an optimization problem is one of the key aspects in SBSE. Therefore, these allocation methods have to be mathematically represented with the optimization problem that each is aiming to solve. Accordingly, the formalization of these methods is the second contribution of this chapter, as many of SSSP approach are missing this point.

Moreover, productivity of resources is the missing feature in all the approaches used in our experiments. Variability of resource's productivity is something that cannot be ignored, and the combination of this feature can provide a clear evidence whether adding this feature would create a complexity level that challenge the team allocation methods, as well as to validate which SSSP approach according to its adopted method can be useful for this particular complexity level. In addition, performing experiments using the five complexity levels on all these methods can provide a validation to the new allocation pattern.

The two main methods that dominate the SSSP approaches are static formation of teams, represented by symmetric assignment, and a dynamic one with arbitrary (asymmetric)

assignment. Symmetric assignment involves two-stages. The first stage is to form the teams by symmetric distribution of resources i.e. overlapping of resources between teams is not permitted, and the resources of each team will continue working together till the end of the project. The second stage is then to allocate those teams to tasks. This type of human resource allocation can be seen in the approaches in [20, 22, 94]. The arbitrary, one on the other hand, assigns multiple resources directly to project tasks regardless of the formation of teams as in [14, 15, 18, 23]. That is each resource can serve in different teams and each task might have a different team from all the others working on it.

Several studies as in [28, 120] show the importance of resources who are participating in a team, to remain in the same team during project time, and completing similar tasks from the beginning till the end of project to maintain their productivity level. That is, sharing developers between different teams and simultaneous tasks has a negative impact on resource's productivity [120, 121]. This is due to the fact that the resources need preparation, knowledge and understanding for any new task in order to complete it. Therefore, moving the resources from one team or task to another requires additional time for the resources to gain the knowledge required for that task. In addition, moving resources from one team to another or sharing a resource across the teams during the course of development as adopted by [14, 15, 18, 28] can cause communication overhead [120]. Cohesion between team members is very important, which mean having them all working in a single team from the beginning till the end of project, makes them less prone to communication overhead that enables them to continue working in harmony and productively [28, 62, 121]. For this reason, controlling the number of team members can be an attempt to reduce the overhead communication and any other negative impacts especially on teams' productivity [28, 121].

On the other hand, many studies such as [15, 118] provide evidence of the dynamic assignment use and practicality in current software organizations. One of the reasons for that is the use of expertise. Expertise of resources is a vital aspect to be addressed especially for the allocation of resources to tasks. Expertise becomes increasingly important to demonstrate the suitability of resources in team allocation, however, sharing the most expert resources across the teams can be to monitor and to ensure the throughput and quality of each team. In addition, changing the team members from one task to another while having multiple tasks with different sizes can lead to better results. For instance, if we have two tasks where one has a very small size and the other is very large one, and both should be performed simultaneously, then having a larger team with suitable expertise doing the large task while the small task has limited number of team members even if they are learning through practice will reduce the overall development and project time.

Therefore, it is important to address the differences between these methods by demonstrating their performance, accuracy, and effectiveness to highlight the benefit(s) and drawbacks of each and in which circumstances they can provide reliable and best solutions. For this reason, it is important to understand how each of these methods is employed by the SSSP approaches. We have found that each one of these two allocation methods can be divided into two. That is, the symmetric assignment can be formed as a static team allocation either for task distribution by queueing system simulator that also simulates the time as in [20-22, 94, 122], or by the allocation of project tasks to the teams with time simulator as in [23]. The asymmetric assignment on the other hand, can be formed as a dynamic team allocation with time simulator, but the participation of each resource in each task can be either with binary participation, which represent either the resource will work on that task or not, or a percentage for participation, which implies that the resource will participate in the task for a percentage of his/her working time.

For simplicity, each allocation method is denoted by the nature of the team allocation including the way of forming and distributing teams, and the project time span simulation used. Therefore, the first method that uses the symmetric assignment providing static teams with queueing simulator to distribute the teams to tasks and simulate project time is given the name Static Teams with Queue Simulator (STQS). The second one is the one that uses symmetric concept and provides static teams but with time simulator of days. This method is given the name Static Teams with Time Simulator (STTS). The third method that uses the asymmetric assignment, as well as binary participation is given the name Dynamic Teams with Binary Participation (DTBP). The fourth one that uses asymmetric assignment but with participation percentage of the resource time is given the name Dynamic Teams with Participation Rate (DTPR). This chapter consequently provides the overall work done on optimizing these methods using GA with standardized settings.

The reminder of this chapter is organized into six sections. Section 5.2 provides the formalization of each team allocation method within the SSSP problem. Section 5.3 details the solution adopted to compare the team allocation methods combined with the GA optimization technique. In Section 5.4, the elements for the experiments are detailed, which include the datasets used, the comparison metrics and measurements adopted, and the experiment results of each method. Section 5.5 discusses the findings, the limitation of this study, and conclude the chapter.

5.2 SSSP Problem Formalization by Four Different Team Allocation Methods

This section provides formalization of the SSSP problem taking into account different team allocation methods. The first subsection of this formalization addresses the main attributes and variables concern the allocation problem in software projects under "General Definition". After this subsection, four team allocation methods are addressed, and each is discussed under a separate title. Each team allocation method subsection provides formalization of the project time span optimization problem with consideration of the constraints, and the software project attributes defined in "General Definitions".

General Definitions:

The problem of staffing the available human resources and scheduling the tasks in a software project (SSSP) can be represented as a software project P that contains a set of tasks denoted by T of size $m, where\ m \in \mathbb{Z}^+$ and the set can be represented as $T = \{t_1, t_2, \dots, t_m\}$. Each $t_i \in T, where\ i = \{1, \dots, m\}$ is characterized by an estimated workload denoted by e_{t_i} in terms of Man-Day effort unit.$\forall\ t_i \in T: e_{t_i} \mapsto e : e \in \mathbb{R}^+$. Since the development of software requires combined skills, and not only technical ones the term competency is used in this problem. Competencies for example means good analytical, logical, and interpretive ability as well as the skill to write a program in a specific language. So, the set of competencies required for developing project tasks or the available resources are possessing is denoted by C of size $u, where\ u \in \mathbb{Z}^+$ and represented as $C = \{c_1, c_2, \dots, c_u\}$.

Moreover, a function TC for each $t_i \in T$ returns the competencies required for t_i as $TC_{t_i} \mapsto c_a : c_a \in C, where\ a = \{1,2, \dots, u\}$. Within the project, there exists dependencies between the tasks so that a task cannot be performed before its predecessors. The set of task dependencies denoted by TD contains m elements representing the number of tasks in T, can be represented as $TD = \{dp_1, dp_2, \dots, dp_m\}$. Each dependency $dp_i \in TD$ represents a dependency between a task and its predecessors, which means that t_i cannot be started until all its predecessors are finished. The value of dp_i however might holds a zero, a single task, or multiple tasks. The dependency dp_i for each t_i maps to a set of tasks denoted by $Z, where\ Z = \{t_{i1}, \dots, t_{iZ}\}$ represented as $dp_i \mapsto Z: Z \subseteq T$.

With set of available resources within the firm to perform the project tasks denoted by R of size $n, where\ n \in \mathbb{Z}^+$, this set can be represented as $R = \{r_1, r_2, \dots, r_n\}$. For each $r_j \in R\ , where\ j = \{1, ,2, \dots, n\}$, the following function denoted by RC returns the competences that r_j possesses by $RC_{r_j} \mapsto c_b: c_b \in C, where\ b = \{1,2, \dots, u\}$. For each $c_s \in C$, each r_j possesses a productivity

denoted by $Pro_{r_j}(c_s)$ measured in term of proficiency level denoted by $pl \in \mathbb{R}^+$: $Pro_{r_j}(c_s) \mapsto pl, where\ 0 < pl \leq 4$.

Static Teams with Queue Simulator Method (STQS)

The allocation of resources is to teams. The set of teams formed for project P is denoted by TM of size $v, where\ v \in \mathbb{Z}^+ \wedge 1 \leq v \leq n$ represented as $TM = \{tm_1, tm_2, \dots, tm_v\}$. The allocation should be performed in two stages. The first stage is to assign the resources to teams and the second one is to assign the formed teams to tasks. For team allocation, the decision variable Q^* returns a Binary value. For each r_j and $tm_o \in TM, where\ o = \{1,2, \dots, v\}$ a value of one means that r_j is assigned to tm_o, and zero otherwise. The decision variable Q^* represented as follow:

$$Q^*(r_j, tm_o) = \begin{cases} 1, & if\ \ r_j\ assigned\ to\ tm_o \\ 0, & Otherwise \end{cases}$$

The assignment of teams to tasks is considered for this problem by a queuing system. This system is a single queue with multi nodes, where each node represents a team, and each package represents a task in the queue waiting to be processed. The set that holds the tasks in the queue denoted by Q contains m elements representing the number of tasks in T, where $Q \subseteq T$ and can be represented as $Q = \{qt_1, qt_2, \dots, qt_m\}$. Each $qt_k \in Q, where\ k = \{1,2, \dots, m\}$ is positioned in the correct order while the system processing the tasks. This position is sorted in the queue according to the dependency constraint $dp_{qt_k} \in TD$. Moreover, the set of processed tasks for all the teams denoted by PT of size v representing the number of teams in TM is depicted by $PT = \{pt_{tm_1}, pt_{tm_2}, \dots, pt_{tm_v}\}$. For each $tm_o \in TM\ \exists\ pt_{tm_o} \in PT \wedge pt_{tm_o} \subseteq Q$ in which stores the tasks that the team has processed containing p elements $where\ p \in \mathbb{Z}^+ \wedge 1 \leq p \leq m$ depicted as $pt_{tm_o} = \{pt_1^*, pt_2^*, \dots, pt_p^*\}$. In this queueing system, the simulation time represents the project time, which counted as the duration of when the first task in the queue is sent to a team for processing untill the last task is finished. However, this representation is equal to the maximum processing time among the teams. Accordingly, each team tm_o has a processing time denoted by $tmTime_{tm_o}$ during the simulation represented as $\forall tm_o \in TM\ \exists\ tmTime_{tm_o}, where$

$$tmTime_{tm_o} = \sum_{w=1}^{p} \frac{e_{pt_w^*}}{\sum_{j=1}^{n} Pro_{r_j}(TC_{pt_w^*}) * \ Q^*(r_j, tm_o)}$$

The set that holds the cumulative processing time for each team is denoted by $timeX$ of size v represented as $timeX = \{tmTime_1, tmTime_2, \dots, tmTime_v\}$. The function that returns project completion time (Time Span) denoted by $f(Time)$ can be represented as $f(Time) = \max\ (timeX)$. $\forall\ qt_k \in Q, r_j \in R, tm_o \in TM$ the problem is to minimize the time span $Time$ of software project P:

$$\min f(Time)$$

Subject to:

- At least one r_j assigned to tm_o should possesses the required competencies for t_i represented as:

$$\forall\, tm_o \in TM, pt_{tm_o} \in PT, pt_p^* \in pt_{tm_o}\ \exists\, r_j.TC_{pt_p} \cap (\sum_{o=1}^{v}\sum_{j=1}^{n} RC_{r_j} * Q^*(r_j, tm_o)) \neq \phi$$

- Number of resource participating in one team should not exceeds 12. Otherwise, penalty will be applied on the solution as a consequence of overhead communication that is anticipated to reduce the team's productivity and the development speed.

$$\forall\, tm_o \in TM, \qquad \sum_{j=1}^{n} Q^*(r_j, tm_o) \leq 12$$

- At least one resource is assigned to each team.

$$\forall\, tm_o \in TM, \qquad R \cap tm_o \neq \phi$$

- The precedence relationship should be met so that for each task in T its predecessors must be finished in order the task to be started.

$$\forall\, t_i \in T\ \exists\, dp_i \in TD: dp_i\ \subseteq PT$$

For this problem representation there is no need for creating a constraint on each team to be assigned to at least one task as the queueing system distributes the tasks in the queue to the first available team.

Static Teams with Time Simulator Method (STTS)

The allocation of resources is to teams. The set of teams formed for project P is denoted by TM of size $v, where\ v \in \mathbb{Z}^+ \wedge 1 \leq v \leq n$ represented as $TM = \{tm_1, tm_2, \dots, tm_v\}$. The allocation should be performed in two stages. The first stage is to assign the resources to teams and the second one is to assign each team to task(s). For team allocation, the decision variable Q^* returns a Binary value. For each r_j and $tm_o \in TM, where\ o = \{1,2, \dots, v\}$ a value of one means that r_j is assigned to tm_o, and zero otherwise. The decision variable Q^* represented as follow:

$$Q^*(r_j, tm_o) = \begin{cases} 1, & if\ \ r_j\ assigned\ to\ tm_o \\ 0, & Otherwise \end{cases}$$

The decision variable Q on the other hand returns a Binary value too, but representing if $t_i \in T$ is allocated to $tm_o \in TM$ with value of one, or zero otherwise as follow:

$$Q(tm_o, t_i) = \begin{cases} 1, & if\ t_i\ is\ allocated\ to\ tm_o \\ 0, & Otherwise \end{cases}$$

The project time span can be estimated by simulating the teams' work over time, so every tick of time representing a day of work is recorded by a variable denoted by $projectTime$ represented as $projectTime = 1,2, \dots, xTime: xTime \in \mathbb{Z}^+$. While the time is ticking, and according to

dependencies between the tasks, some of these tasks are in progress, some are waiting, and others might be finished. The first set that stores the operating tasks denoted by RT of size m, represented as $RT = \{rt_1, rt_2, \dots, rt_m\}$. The value for each $rt_p \in RT, where\ p = \{1,2, \dots, m\}$ is a binary, representing whether rt_p is currently in progress by value of one, or not with value of zero. Similarly, the set of tasks that are finished denoted by FT represented as $FT = \{ft_1, ft_2, \dots, ft_m\}$. The value of each $ft_p \in FT$ is also represented by a binary value, which indicate whether ft_p is finished having value of one, or zero otherwise.

For each $projectTime$ and a given t_i assigned to tm_o that has a value of one in RT, the estimated effort e_{t_i} on each $projectTime$ is reduced by the sum of each resource productivity $Pro_{r_j}(TC_{t_i})$ associated with the competency required for the task t_i, and assigned to tm_o, represented as:

$$\forall\ projectTime, t_i \in T:$$

$$e^*_{ti} = \left(e_{t_i} - \left(\sum_{tm \in Tm} \sum_{r \in R} Pro_{r_j}(TC_{t_i}) * Q^*(r_j, tm_o) * Q(tm_o, t_i) * rt_i \right) \right), where\ e^*_{ti} = \{e^*_{t1}, \dots, e^*_{tm}\}$$

This loop continues until e^*_{ti} converge to zero, and every element in FT has the value of one. Then at this stage, the function that returns project completion time (Time Span) denoted by $f(Time)$ can be represented as:

$$f(Time) = xTime \Leftrightarrow \forall t_i \in T: e_{t_i} \approx 0 \wedge \sum_{i=1}^{m} ft_i = m$$

$\forall\ t_i \in T, r_j \in R, tm_o \in TM,$ The problem is to minimize the time span $Time$ of software project P as follow:

$$\min f(Time)$$

Subject to:

- At least one r_j assigned to tm_o should possesses the required competencies for t_i represented as:

$$\forall\ t_i \in T, \exists\ r_j . TC_{t_i} \cap \left(\sum_{o=1}^{v} \sum_{j=1}^{n} RC_{r_j} * Q(tm_o, t_i) * Q^*(r_j, tm_o) \right) \neq \phi$$

- Number of resource participating in one team should not exceeds 12. Otherwise, penalty will be applied on the solution as a consequence of overhead communication that is anticipated to reduce the team's productivity and the development speed.

$$\forall\ tm_o \in TM, \qquad \sum_{j=1}^{n} Q^*(r_j, tm_o) \leq 12$$

- At least one resource is assigned to each team.

$$\forall\ tm_o \in TM, \qquad R \cap tm_o \neq \phi$$

- Each team has to be assigned at least to one task.

$$\forall\, tm_o \in TM, t_i \in T: \exists \sum_{i=1}^{m} Q(tm_o, t_i) \geq 1$$

- The precedence relationship should be met so that for each task in T its predecessors must be finished in order the task to be started.

$$\forall\, t_i \in T\ \exists\, dp_i \in TD: dp_i\ \subseteq FT$$

Dynamic Teams with Binary Participation Method (DTBP)

The allocation of resources is to tasks performed by assigning each resource to a set of tasks. For this resource allocation, the decision variable Q returns a Binary value for each $r_j\ \in R$, and $t_i \in T$, where a value of one means that r_j is assigned to t_i, and zero otherwise. The decision variable Q represented as follow:

$$Q(r_j, t_i) = \begin{cases} 1, & if\ r_j\ is\ allocated\ to\ t_i \\ 0, & Otherwise \end{cases}$$

The project time span $Time$ can be calculated by simulating the work of resources on tasks as a dynamic teams over time, so every tick of time is representing a day of work recorded by the variable denoted by $projectTime$, and represented as: $projectTime = 1,2, \dots, xTime: xTime \in \mathbb{Z}^+$. While the time is ticking, and according to dependencies between the tasks, some of these tasks are in progress, some are waiting, and others might be finished. The first set that stores the operating tasks denoted by RT of size m is represented by: $RT = \{rt_1, rt_2, \dots, rt_m\}$. The value for each $rt_p \in RT, where\ p = \{1,2, \dots, m\}$ is a binary, representing whether rt_p is currently in progress by value of one, or zero otherwise. Similarly, the set of tasks that are finished denoted by FT can represented as: $FT = \{ft_1, ft_2, \dots, ft_m\}$. The value of each $ft_p \in FT$ is also represented by a binary value, which indicate whether ft_p is finished having value of one, or zero otherwise.

For each $projectTime$, t_i that has a value of one in RT, and r_j assigned to t_i such that $Q(r_j, t_i) = 1$, the estimated effort e_{t_i} on each $projectTime$ is reduced by the sum of the assigned resources' productivity $Pro_{r_j}(TC_{t_i})$ associated with the competency required for task t_i, represented as:

$$\forall\, projectTime\ \wedge\ t_i \in T: rt_{t_i} = 1$$

$$e^*_{ti} = \left(e_{t_i} - \left(\sum_{j=1}^{n} Pro_{r_j}(TC_{t_i}) * Q(r_j, t_i) * rt_i \right) \right), where\ e^*_{ti} = \{e^*_{t1}, \dots, e^*_{tm}\}$$

This loop goes until e^*_{ti} converge to zero, and every element in FT has the value of one. Then at this stage, the function that returns project completion time (Time Span) denoted by $f(Time)$ can be represented as:

$$f(Time) = xTime \Leftrightarrow \forall t_i \in T: e_{t_i} \approx 0 \wedge \sum_{i=1}^{m} ft_i = m$$

$\forall\, t_i \in T, r_j \in R$, The problem is to minimize the time span of project P:

$$\min f(Time)$$

Subject to:

- At least one r_j assigned to t_i should possesses the required competencies for t_i represented as:

$$\forall\, t_i \in T, \exists\, r_j: TC_{t_i} \cap (\sum_{j=1}^{n} RC_{r_j} * Q(r_j, t_i)) \neq \phi$$

- Number of resources participating to perform one task should not exceeds 12. Otherwise, penalty will be applied on the solution as a consequence of overhead communication that is anticipated to reduce the team's productivity and the development speed.

$$\forall\, t_i \in T: \sum_{j=1}^{n} Q(r_j, t_i) \leq 12$$

- At least one resource is assigned to each task.

$$\forall\, t_i \in T: \sum_{j=1}^{n} Q(r_j, t_i) \geq 1$$

- Each resource has to be assigned to at least one task.

$$\forall\, r_j \in R: \sum_{i=1}^{m} Q(r_j, t_i) \geq 1$$

- The precedence relationship should be met so that for each task in T its predecessors must be finished in order the task to be started.

$$\forall\, t_i \in T\ \exists\, dp_i \in TD: dp_i \subseteq FT$$

Dynamic Teams with Participation Rate Method (DTPR)

The allocation of resources to tasks is performed by assigning each resource to a set of tasks. However, this allocation associates a percentage to represent the amount of participation of each resource in each task denoted by pr, where $pr = \{0, 0.25, 0.5, 0.75, 1\}$. For each $r_j \in R$, and $t_i \in T$, the variable Q returns the participation percentage value associated with the assignment of resource r_j to task t_i representing whether this resource will participate in this task with an amount of its working time, or not. The variable Q represented as follow:

$$Q(r_j, t_i) = \begin{cases} 1, & if\ r_j\ assigned\ 100\%\ to\ t_i \\ 0.75, & if\ r_j\ assigned\ 75\%\ to\ t_i \\ 0.5, & if\ r_j\ assigned\ 50\%\ to\ t_i \\ 0.25, & if\ r_j\ assigned\ 25\%\ to\ t_i \\ 0, & Otherwise \end{cases}$$

The project time span can be calculated by simulating the work of resources on tasks as dynamic teams over time, so every tick of time is representing a day of work recorded by the variable denoted by $projectTime$, represented as: $projectTime = 1,2, \dots, xTime: xTime \in \mathbb{Z}^+$. While the time is ticking, and according to dependencies between the tasks, some of these tasks are in progress, some are waiting, and others might be finished. The first set that stores the operating tasks denoted by RT of size m, is represented by: $RT = \{rt_1, rt_2, \dots, rt_m\}$. The value for each $rt_p \in RT, where\ p = \{1,2, \dots, m\}$ is a binary, representing whether rt_p is currently in progress by value of one, or zero otherwise. Similarly, the set of tasks that are finished denoted by FT can represented as: $FT = \{ft_1, ft_2, \dots, ft_m\}$. The value of each $ft_p \in FT$ is also represented by a binary value, which indicate whether ft_p is finished having value of one, or zero otherwise.

For each $projectTime$, t_i that has a value of one in RT, and r_j assigned to t_i such that $Q(r_j, t_i) > 0$, the estimated effort e_{t_i} on each $projectTime$ is reduced by the sum of the assigned resources' productivity $Pro_{r_j}(TC_{t_i})$ associated with the competency required for task t_i multiplied by the participation percentage of each, represented as:

$$\forall\, projectTime\ \wedge\ t_i \in T: rt_{t_i} = 1$$

$$e^*_{ti} = \left(e_{t_i} - \left(\sum_{j=1}^{n} Pro_{r_j}(TC_{t_i}) * Q(r_j, t_i) * rt_i \right) \right), where\ e^*_{ti} = \{e^*_{t1}, \dots, e^*_{tm}\}$$

This loop continues until e^*_{ti} converge to zero, and every element in FT has the value of one. Then at this stage, the function that returns project completion time (Time Span) denoted by $f(Time)$ can be represented as:

$$f(Time) = xTime \Leftrightarrow \forall t_i \in T: e_{t_i} \approx 0 \wedge \sum_{i=1}^{m} ft_i = m$$

$\forall\, t_i \in T, r_j \in R,$ The problem is to minimize the time span of project P:

$$\min f(Time)$$

Subject to:

- At least one r_j assigned to t_i should possesses the required competencies for t_i represented as:

$$\forall\, t_i \in T, \exists\, r_j: TC_{t_i} \cap \left(\sum_{j=1}^{n} RC_{r_j} * Q(r_j, t_i) \right) \neq \phi$$

- Number of resources participating to perform one task should not exceeds 12. Otherwise, penalty will be applied on the solution as a consequence of overhead communication that is anticipated to reduce the team's productivity and the development speed.

$$\forall\, t_i \in T, : \sum_{j=1}^{n} Q(r_j, t_i) \leq 12$$

- At least one resource is assigned to each task.

$$\forall\, t_i \in T\colon \sum_{j=1}^{n} Q(r_j, t_i) \geq 1$$

- Each resource has to be assigned to at least one task.

$$\forall\, r_j \in R, t_i \in T\colon \exists \sum_{i=1}^{m} Q(r_j, t_i) \geq 1$$

- The precedence relationship should be met so that for each task in T its predecessors must be finished in order the task to be started.

$$\forall\, t_i \in T\ \exists\ dp_i \in TD\colon dp_i\ \subseteq FT$$

It can be seen from the description above that the definition of the four methods is partially similar to each other. However, the differences exists in these problems are vital for the solution representation, which will be used by the optimization techniques. The following section provides the solutions to each method and their optimization process details.

5.3 Genetic Algorithm Configurations and Operators Solution

As an NP-Hard problem complexity, Meta-Heuristic techniques can be used to approximate a solution for the SSSP problem. One of the most used and powerful among the Meta-Heuristic techniques is the Genetic Algorithm (GA) [5] proposed by John Henry Holland in [83]. This algorithm has been employed for optimizing various software engineering problems [5]. The algorithm develops a solution based on the principle of life evolution, and natural selection of genes. The heuristics in this algorithm are designated in two operations, which are crossover, and mutation. The main parts that should be considered while using this algorithm to solve the SSSP problem can be illustrated from the work of [18, 21, 22]. These parts are the solution representation and encoding mechanism, initial population for the solution, stochastic operations, objective function containing the team allocation method, commitment of resources, and scheduling technique for fitness selection, and the optimizer settings proposed. Most of the approaches that propose a solution for the SSSP using a Meta-Heuristic technique follow these parts in their discussion.

We have employed this algorithm to implement each allocation method -described previously in Section 5.2- combined with a fitness function that simulates project time with consideration of dependencies between project tasks, and resources' competencies and productivity. The description of the GAs, and their main parts of the optimization process are depicted in the following sections.

5.3.1 Solution Representation and Chromosome Encoding

The solution representation for an optimization problem encompasses a solution structure and its possible encoding system for the problem elements [123]. The solution structure can be represented by a one (1D, or vector), two (2D), or a multi-dimensional (ND) matrix. The basic solution structure in GA is a vector chromosome. Each element in the chromosome structure is called gene, and the content of this gene is called an allele [86]. The values of a chromosome can be encoded using different encoding systems. The encoding of a chromosome can be in a binary, permutations, value, or tree structure of genotype [86].

The solution representation of the first team allocation method named "Static Teams with Queue Simulator" (STQS) is illustrated from the approaches in [20-22, 94]. This method uses two vector chromosomes. The first chromosome represents the resources and their distribution into teams, and the second one represents the tasks and their order in a single queue system. The representations of both chromosomes having n number of resources, and m number of tasks are depicted in the following Figure 21.

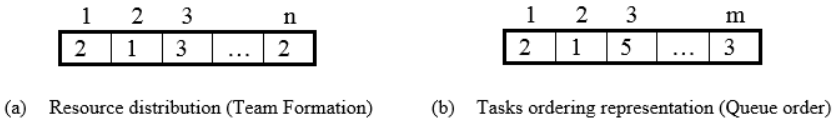

Figure 21: STQS method Chromosomes

The first chromosome - in Figure 21(a) – represents the resources distribution solution, where each gene represents a resource, and each allele represents a team. That is, the value for each resource (gene) holds the team number that this resource is assigned to. The encoding system used for this chromosome is represented by an integer value encoding. The distribution of resources into teams should be performed in this chromosome according to the pigeonhole principle, which as a key aspect allows more than one resource to have the same team number in the solution (multiple genes have the same allele value) [124]. For instance, resource 1 and n in the figure both work in team 2.

The second chromosome depicted in Figure 21(b) provides the ordering solution representation that show the execution order for each project task i.e. to where each task should be sorted in the queue. Each gene in this chromosome represents a task, and each allele represents the task order in the queue. Unlike the previous chromosome, the ordering representation do not allow similarity between allele values. Therefore, the encoding system used in this solution representation to

comply with this restriction is a permutation encoding. For example, if we have three elements representing the project tasks, the encoding of these tasks for their order within the queue using the permutation encoding can be 1, 2, 3, or 2, 1, 3, or 3, 1, 2, etc.

The second resource allocation method defined in this chapter is "Static Teams with Time Simulator" (STTS). The solution representation for this method is depicted in Figure 22 by two chromosomes. The structure of those chromosomes is a vector. The first chromosome similar to the one in STQS method represents the distribution of resources into teams. Unlike STQS method representation, the second chromosome in this method solution represents the task allocation by providing the team number that is responsible to work on each task. This representation of STTS method solution having n number of resources, and m number of tasks is depicted as follow.

(a) Resource representation (Team Formation) (b) Teams to tasks representation (Team Allocation)

Figure 22: STTS Method Chromosomes

From the previous Figure 22, it can be seen that the pigeonhole principle again is the one that forms the solution for both chromosomes. For the first chromosome representation (a), the resources are distributed into teams, where each gene represents a resource, and each allele represents a team. This representation provides to which team each resource is assigned to. Representation (b) on the other hand, depicts the distribution of teams to tasks. From this representation, it can be seen that which task should be done by which team. For example, from the Figure 22(b), we can see that task 2 is assigned to team 1, and task 3 is assigned to team 3.

The third method of "Dynamic Teams with Binary Participation" (DTBP) depicted in the following Figure 23 represents the solution by a 2-D matrix structure. The vertical dimension (columns) represents the resources, and the horizontal one (rows) represents project tasks. This representation having n number of resources, and m number of tasks is depicted as follow.

	1	2	3	…	n
1	1	0	1	…	1
2	0	0	1	…	0
3	0	1	0	…	1
:	0	1	1	…	1
m	1	1	0	…	1

Figure 23: DTBP Method Chromosome

From Figure 23, it can be seen that this representation assigns each resource with a high probability to serve in different teams during project time. For instance, it can be seen in Figure 23 that resource 1 assigned with resources 3, and n to work on tasks 1, where on task m this resource works with resources 2, and n. We have defined this representation as an arbitrary assignment of resources to tasks. Each gene in this chromosome has two positioning points (v and h) that define the resource (v) and the task (h) that (s)he assigned to. In addition, the allele of each gene should be encoded using the binary system, which implies whether resource (v) is assigned to task (h) by value of one, or zero otherwise. This representation moreover, requires important assignment constraint in order to gain realistic and reasonable solution. For instance, this representation can provide a solution where all resources are assigned to all tasks. For this reason, a constraint is implemented with this representation to make sure that any resource works on more than one task at a time, its productivity will be normalized to the number of these simultaneous tasks. By doing so, the solution then of having all the resources works on all the tasks will provide low quality solution of project time span.

The fourth method of "Dynamic Teams with Participation Rate" (DTPR) represents the solution by 2-D structure similar to DTBP. This representation having n number of resources, and m number of tasks is depicted in the following Figure 24.

	1	2	3	…	n
1	0.25	0.5	0.75	…	0
2	0	0	1	…	0.5
3	0	0.5	0.25	…	1
:	0.75	1	1	…	0.25
m	1	0.25	0	…	0.5

Figure 24: DTPR Method Chromosome

From Figure 24, it can be seen that this assignment representation is similar to the one in DTBP method. However, this method enforces the resources to partially dedicate a percentage of their working time to each task to which they are assigned. With five different values described in the problem formalization Section 5.2 of this method, each allele in the chromosome representing the gene (v, h) of resource (v) and task (h) can hold a real number value encoding from the range {0, 0.25, 0.5, 0.75, 1}. It is noteworthy that the same constraint described for the pervious method DTBP is adopted in this method implementation too. This is to make sure that a full dedication of all the resources to all tasks, implying overtime work assignment is not considered.

5.3.2 Initial Population

Two methods can be used to create an initial population for a solution using GA. The first one is random initialization, which populates the solution randomly. The second one is heuristic initialization that uses one of the heuristic techniques such as Greedy, Hill Climbing, etc. -see Section 2.2- to create an initial population [97]. For a diversity of solutions within the initial population, the random initialization can provide better results of optimality than the heuristic one for two reasons. Heuristic methods can lead for initializing redundant individuals over the population, which will lead to less diversity of solutions. In addition, GA has a selection operation to heuristically create new population(s).

The initial population used for all the methods is the random initialization with value encoding either for team formation, queue order, team allocation, or both 2-D matrix chromosome creation. For more information about this particular GA aspect, see [97] [86, 123].

5.3.3 Crossover Operator

One of the stochastic operations in GA is crossover. This operation creates new solutions by exchanging subparts of two single chromosomes to create two new ones mimicking the biological combination of parents' chromosomes into new child chromosomes [86]. Crossover can be implemented using different operators that identify which subparts are to be selected for the new child and how they will be combined into one. These crossover operations can be a single-point, two-point, uniform, and arithmetic process [86, 97, 102].

Many of the SSSP approaches have modified their own crossover operation according to their representation of the problem, and the solution. While two different chromosome structures are used by the team allocation methods, we have formed two types of crossover each to fit with the corresponding method for the solution structure.

The chromosomes used for STQS and STTS methods are modified for the experiments to be in a single chromosome representation combining both resources, and tasks. This combination of two chromosomes into one is inspired by the work of [22], which can speed up the selection of fitted solutions by the objective function. The crossover operation starts by dividing the single chromosome into two parts, based on the number of project tasks (m) and the resources assigned to it (n) depicted by the example of five resources, and five tasks in the following Figure 25.

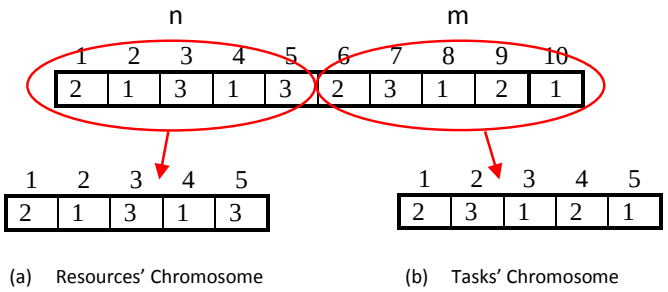

Figure 25: STQS, and STTS methods Chromosome separation

As can be seen from Figure 25 above, according to the number of resources (n=5) and tasks (m=5), the single chromosome is separated into two having five resources, and five tasks. Unlike the operator used in [22], those two new chromosomes are stochastically modified by two-point crossover. However, we have modified this crossover to apply inversion operation on the values of the area defined between the two random points.

Employing two parents in crossover operation for those particular allocation methods, unless a constraint checker is exists, can lead to an invalid solution(s). It is important to notice that the random creation of solutions in the population for those allocation methods can create a solution that holds a number of teams different from any other ones. For instance, the first solution in the population might have 4 teams that the resources have been distributed to and assigned to different tasks, whereas another solution could have only two teams. If we apply crossover on those two solutions having them as parents, then there might be one of the tasks in the first solution that has been assigned to a team that no longer exists after the crossover. Consequently, applying two point crossover with two parents to create a child for those two allocation methods might produce an invalid solution(s). Therefore, a modified operator has been created to ensure that none of the resources nor the teams are left with an invalid assignment. The process to crossover both chromosomes depicted by the example in the following Figure 26.

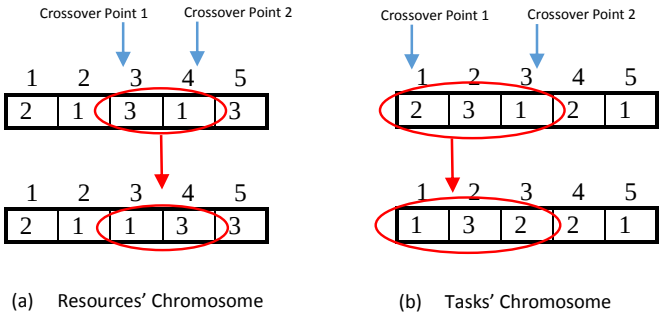

Figure 26: STQS, and STTS methods Crossover

From Figure 26, it can be seen that both chromosomes have a subpart of their solutions changed. As the operator adopted for both allocation methods is a 2-points crossover, the area between those two random points is the one that the crossover will be applied to. In chromosome (a) for instance, the crossover area selected is between genes 3 to 4 to swap their values. The crossover operation has swapped the resources 3 and 4 to teams 1 and 3 respectively.

On the other hand, the crossover operation applied on "tasks' chromosome" depicted in Figure 26(b) has selected a random area between tasks 1 to 3. The resulting operation assigned task 1 to team 1, and task 3 to team 2. Noteworthy, this operation has left task 2 with the same assignment to team 3, as our proposed crossover is to invert the values for the area defined between two random points. It is important to notice that the previous two figures representing crossover of both STQS and STTS, provides examples on how the operation can be performed but not to provide the exact operation of both. STQS differ from STTS by the task chromosome representation, which distributes the task orders for STQS and the one in STTS distribute the tasks on teams.

The second crossover proposed is a 2-points crossover operator for 2-D matrix, which can be applied on DTBP and DTPR methods as both have the same chromosome structure. This operation selects two random point for each dimension of the 2-D matrix depicted by the example in the following Figure 27.

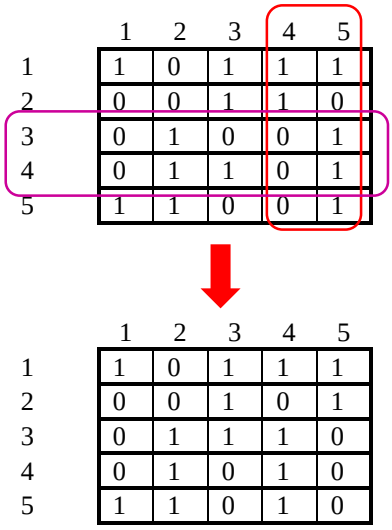

Figure 27: DTBP, and DTPR methods Crossover

In Figure 27, the vertical dimension represents the resources, and the horizontal one represents the tasks. It can be seen from the figure that two points for each dimension are selected by the crossover operator. The crossover selected area for the resource dimension is between resources

4, and 5. On the other hand, the crossover selected area for the task dimension is between tasks 3, and 4. The crossover operator has inversely changed the values of each area similar to the methods used for STQS and STTS. For instance, the assignment values have been swapped between resources 5 and 4 by applying the crossover on the resource dimension. On the other hand, the assignment values of the task dimension have been swapped between tasks 3 and 4 by the same crossover operation.

5.3.4 Mutation Operator

Mutation in GA, as one of the stochastic operations, involves random changes on the solution chromosome generated. Mutation operator is also used to avoid the generation of same solutions, by which it can lead for more exploration in the solution space for an optimal or near optimal result [86]. That is, the resulting chromosome by mutation operation can move the search to a global area in the solution space [103]. This operation is usually applied with a low probability for creating diversity of chromosomes in the population of GA [97]. Moreover, many approaches such as the work of [15] have employed the 1+1 EA, which merely use the mutation, and eliminate the crossover from the approach.

There are different ways of mutating a chromosome. One way can be by randomly generating a new value for a random number of bits, another one is by flipping the bits of the chromosome [97]. Additional to those mutations, there are five operators, which are bit flip or random resetting of single gene, swapping of two points, scrambling a part of the chromosome for permutation encoding, and inversion for a part of the chromosome to flip it [102]. While mutation is related to the process part in GA for exploration of search space [103], we have increased the rate of mutation for the experiments on the team allocation methods to explore a wider area of the solution space for global optima exploration.

For STQS and STTS methods, we have developed a mutation process that starts with dividing the chromosome into two parts as both methods combine two representations into a single chromosome. Each part from this division then forms a chromosome that contains the representation of either the resources, or tasks. Unlike the crossover of both methods at this point the employed mutation operator for the STQS method differs from the STTS one. Mutation used for the STQS method for both chromosomes is the swap mutation. This mutation randomly selects two genes and exchange their values. Mutation for the STTS method chromosomes on the other hand, performs a modified mutation on two random genes. Our modified mutation after the separation of the single chromosome into two, works as follow:

- For each chromosome, two genes should be selected.
- Starting by the resources' chromosome, a new value for each selected gene should randomly be generated according to the maximum team number (gene value) exists within the chromosome.
- After mutating the resources chromosome, the operator moves to the tasks chromosome, and stores the new maximum team number value of the resource chromosome.
- The operator then checks whether any gene has a team number that does not exists in the mutated resources chromosome, if so:
  - The chromosome reseeded those genes with a new random team number according to the new maximum team value. or
  - The operator continues generating a new team for each selected gene.

This modified mutation operator is depicted in the following Figure 28.

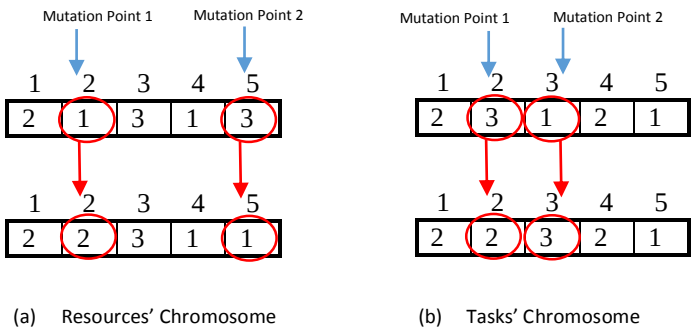

Figure 28: STTS method Chromosome Mutation

As can be seen in the Figure 28, two random genes in each solution representation chromosome are selected. In Figure 28 (a), gene 2, and 5 are selected, where in Figure 28 (b), gene 2, and 3 are selected. Applying mutation on these selected genes has made each having a new random value. For instance, after applying mutation on gene 5 in chromosome (a) its value changed from 3 to 1, and for gene 3 in chromosome (b) its value changed from 1 to 3. These generated values are representing the team number in both representations. It is noteworthy that the mutation process has continued on chromosome (b) to mutate both selected genes with same consideration of the maximum number of teams left from the resources chromosome mutation. This can be seen by mutation of gene 3 in chromosome (b), in which the value of this gene has changed to 3 as this team value still exists after chromosome (a) mutation.

We also propose a modified mutation for both DTBP and DTPR methods as their solution is represented by a 2-D matrix chromosome. This modified mutation selects two rows, and two columns to mutate the 2-D chromosome. The operator starts by selecting two random columns

according to the number of resources, and two random rows according to the number of tasks. Each gene exists within the selected columns and rows will be individually mutated according to the assignment value associated for the targeted method. For example, if this chromosome represents the DTBP method, then each gene will be either mutated from 0 to 1, or vice versa. For DTPR method however, mutation will randomly select a value for the gene from the range {0, 0.25, 0.5, 0.75, 1}. It is important to notice that using mutation on this representation can lead to have the values for the whole selected rows or columns within the chromosome similar to the values prior mutation. Therefore, our modified mutation for both methods' chromosomes eliminates the previous value of the genes from the random generation of new ones.

5.3.5 Selection Operator

The selection as one of the heuristic operations in GA selects the fittest chromosomes according to their solution fitness value to the objective function to be used on producing a new population(s). Selection can be performed using different methods [86]. These methods are roulette wheel, stochastic universal sampling, tournament, steady state, rank, elitism, and random selections [86]. It is crucial to define the best selection process for a successful GA [97]. That is the selection process can either lead for an optimal or near optimal results by good chromosomes' "solution" diversity, or undesirable solution of the known "premature convergence" by dominating of one extremely fit solution over the entire population. The implementation of the team allocation methods was performed using Matlab 2016, with Global optimization toolbox. The available selection types in Matlab are five. These types are stochastic uniform, remainder, uniform, roulette, and tournament selections. However, two selection operators are mainly used by many SSSP approaches as in [20, 22], which are the roulette wheel, and tournament selections. The one that has been used in all our experiments for this chapter is the roulette wheel selection. For more details and description about these operators, the reader can refer to [86, 97].

5.3.6 Fitness Function

The objective function defined for the SSSP problem in Section 1.2 is a cost function searching for the most minimized solution of software project time span. As the team allocation methods optimization problem is part of SSSP, the fitness function developed to test these methods involves simulation of project time. Simulating project time within the fitness function of an optimization technique can be performed using different simulation models. Two main time simulators however are developed each of which can be used for specific team allocation method(s). The first simulates project time for a queueing system that has a queue of project tasks

that need to be served by different teams, so the time consumed to serve all the tasks is the estimated project time span similar to what proposed by [20, 125]. The other one simulates project time as the project progress while the assigned resources performing project tasks i.e. the time is counted according to a counter and the end value of this counter is the estimated project time span.

The fitness function used for the STQS method is adopted from [20, 22, 125]. This function simulates the development as a queueing system to estimate project time span. This system has a queue that holds the tasks to be done, and the teams that are servers to do the service for each queue element (task). It is worth mentioning that the work in [22] provides the time estimate of a software project according to the queue time. The queue time starts when the first package in the queue is despatched, and ends when the last package is completed. Therefore, their time estimate depends on the definition of start and end time for each package in the queue. So, the end time of the last package is the project time span. Unlike the time estimate provided in these approaches, project time span considered by the fitness function developed for the team allocation method STQS is the longest team time among the teams (servers). The description of the fitness function of project time simulation is represented by the algorithm depicted in the following Figure 29.

1	***Function fitnessValue = QueueSimulator (Solution, R, T)***
2	*Let* $\boldsymbol{Capa} = \mathbf{0}$, *a variable to hold productivity of the team assigned to the current task*
3	*Let* ***Solution*** *be the allocation solution generated by the optimization*
4	*Let* ***R*** *be the resources' set*
5	*Let* ***T*** *be the project tasks' set of size m*
6	*Let* ***TF*** *be an empty set to hold the resources in each team*
7	***for*** *each resource r in Resources*
8	$TF\left(solution(r)\right).add\ (r)$
9	***end*** *for*
10	*Let* ***teamTime*** *be an empty set to hold the time spent by each team*
11	*Let* ***FinTasks*** *be an empty set to hold the finished tasks*
12	*Let* $\boldsymbol{CurrentT} = t_1$
13	*Let* $\boldsymbol{idx} = 1$
14	***while*** $\sum FinTasks < m$
15	$idx = idx(\min(teamTime))$
16	***if*** $dp_{CurrentT} = 0 \,||all\ in\ dp_{CurrentT} \subseteq FinTasks$
17	***for*** *each resources* r_j *in* TF_{idx}
18	$Capa = +\sum pro_{r_j}(TC_{CurrentT})$
19	***end*** *for*
20	$teamTime_{idx} = +\left(\frac{e_{CurrentT}}{Capa}\right)$
21	*remove CurrentT from T*
22	$FinTasks_{CurrentT} = 1$
23	***else***
24	*send CurrentT back of the queue*
25	***end*** *if*
26	***end*** *while*
27	$fitnessValue = \max(teamTime)$
28	***end*** *Function*

Figure 29: Queueing Simulator Fitness Function

It can be seen from Figure 29 that the fitness function requires three inputs. These inputs should provide information about project human resources (R), project tasks (T), and the heuristic solution ($Solution$) generated by the GA to measure its fitness. According to these inputs, the simulator can then sort the resources into teams and store them in a TF set throughout the lines 7 to 9. Noteworthy that TF is a cell array, and the part of the *solution* involved in defining the teams is the first part of the solution (chromosome) that represents the resource assignment to

teams. Based on the formation of teams, the fitness function will be able to estimate the time span of each task.

The first part of the task time span estimation is depicted in the lines 17 to 19 in Figure 29. This part involves determining the cumulative productivity ($Capa$) of the available team TF_{idx} who will perform the current task $CurrentT$ depicted by line 18. This productivity is calculated by the summation of each resource's productivity pro_{r_j} assigned to that team. Resource's productivity however is represented by different competencies, and it can be retrieved according to the competency required for the current task $CurrentT$ represented by $TC_{CurrentT}$.

Based on team's productivity, the estimation of task time span can be calculated by the division of the estimated effort of the current task $e_{CurrentT}$ over the productivity ($Capa$) of the assigned team. This value will be added to the team's time matrix $teamTime_{idx}$ for the corresponding available team index idx depicted by line 20. Once the calculation is finished, $CurrentT$ will then be removed from the task queue T depicted by line 21, and in line 22 this task will be recorded as finished in the ($FinTasks_{CurrentT}$) matrix. The simulator in this fitness function keeps tracking dependency between tasks throughout the simulation of the queue system depicted in line 16. Once the precede task(s) in $dp_{CurrentT}$ for $CurrentT$ are recorded in $FinTasks_{CurrentT}$ matrix, ($CurrentT$) can be then proceed to the next available team TF_{idx}. The identification of a team's availability is depicted in line 15 as checking the least team's time among the set $teamTime$ and recorded the first least working time among the teams in idx. This simulation last as the summation of the binary array $FinTasks$ is less than the number of project tasks. Once all the tasks are finished, which implies that the sum of $FinTasks$ equals m, the return value of the fitness function is the maximum team time among the set $teamTime$.

The second fitness function proposed for simulating the project time span of the team allocation methods STTS, DTBP, and DTPR is depicted in the following Figure 30. However, the solution generated for "resources to teams" assignment, and "teams to tasks" allocation by the GA for each method differs from one method to another. The solution of STTS method combines two representations into a single array. This array holds the information regarding the resources and their distribution to teams represented by the first part of the array, and the team allocation to each task represented by the second part. At the beginning of the fitness function this array will be divided into the original parts corresponding to each representation of resources and tasks. On the other hand, the solution structure of both DTBP and DTPR is 2-D matrix. This solution holds the representation of resource in the vertical dimension, and their assignment to tasks either for binary or percentage to the tasks in the horizontal dimension. The solution generated by the GA are named $Solution$ and after its divisions for the STTS method, the part that represents the

resources' distribution is named $SolutionR$, and the part that represents team distribution to tasks is named $SolutionT$.

The information held by the GA solution are then used by the fitness function simulation model to calculate project time. This model simulates project schedule in terms of days, and considers four vectors. These vectors are depicted in Figure 30 by four sets. The first set (primaryTasks) should initially hold all project tasks, and continue to hold those task(s) that are still waiting to be performed. The second one (unlockedTasks) holds the tasks that have no dependency constraints or those task(s) where their predecessor(s) as the time progress are completed. The third one (operatingTasks) holds the tasks that are under development. And finally the fourth one (finishedTasks) holds the tasks that are completed.

At the beginning of the simulation, the set (primaryTasks) will hold all project tasks, and (finishedTasks) must be an empty set. Any task in (primaryTasks) that has no dependency constraint will be moved then to (unlockedTasks) depicted in the figure by lines from 9 to 14. The simulation accordingly starts by moving the task(s) in (unlockedTasks) to the set (operatingTasks) so that all the tasks in (operatingTasks) can be performed at the same time depicted by line 17. For each task in this set, the associated productivity to its required skill(s) possessed by each resource assigned to it $pro_r(TC_{t_a})$ will be stored in $Capa$ variable that represents the resources capability depicted by line 21. However, the way of determining the value of this variable differs from one allocation method to another.

A. For STTS method, $Capa$ value will be determined by first identifying from $SolutionT$ the allocated team to t_a, and then for each resource r assigned to this team exposed in $SolutionR$ his/her $pro_r(TC_{t_a})$ will be stored in $Capa$.
B. For DTBP method, $Capa$ value will be determined as depicted in line 21 in the figure. However, all the resources assigned to the task t_a will be identified by the vertical lines that have the value of 1 corresponding to the horizontal line of t_a.
C. For DTPR method, $Capa$ value will be determined by multiplying $pro_r(TC_{t_a})$ by the participation rate defined for resource r in the GA solution to the corresponding task t_a in the horizontal line represented by ($Solution(r, t_a)$). The resources assigned to t_a can be identified if $Solution(r, t_a) > 0$.

```
1  Let Capa = 0, a variable that holds capability of all the resource
       assigned to a task
2  Let Solution be a heuristic GA solution
3  Let primaryTasks = T, a set that holds all project tasks
4  Let unlockedTasks be
       an empty set to hold the tasks that are ready to perform
5  Let operatingTasks be an empty set to hold the tasks in progress
6  Let finishedTasks be an empty set to hold the finished tasks
7  Let xTime = 0, the variable that will represents project time
8  Let Scnt be a variable to hold the amount of
       simulaneous tasks a resource at a time is performing
9  for all tasks t_i in primaryTasks
10     if dp_{t_i} = 0
11        remove t_i from primaryTasks
12        add t_i to unlockedTasks
13     end if
14 end for
15 while ∑ finishedTasks < m
16     xTime = +1
17     operatingTasks = unlockededTasks
18     for each t_a in operatingTasks
19        Scnt = 0
20           for each resources r assigned to t_a
21              Capa = + ∑ pro_r(TC_{t_a})
22              for each t_u in operatingTasks
23                 if r assigned to t_u
24                    Scnt = +1
25                 end if
26              end for
27           end for
28           Capa = Capa / Scnt
29           e_{t_a} = e_{t_a} − Capa
30           if e_{t_a} ≈ 0
31              remove t_a from operatingTasks
32              remove t_a from unlockedTasks
33              finishedTasks_{t_a} = 1
34           end if
35           for all tasks t_w in primaryTasks do
36              for all tasks t_z in dp_a do
37                 if finishedTasks_{dp_a} = 1
38                    add t_w to unlockedTasks
39                 end if
40              end for
41           end for
42     end for
43 end while
```

Figure 30: Time Simulator Fitness Function

The value in $Capa$ moreover will be normalized according to the variable$Scnt$, which holds the number of simultaneous tasks that each resource is working on in (operatingTasks) depicted by the lines from 22 to 28. The estimated effort e of task $t_a$will be according reduced by the normalized $Capa$ value depicted by line 29. Once e_{t_a} converges to zero, task t_a will be removed from both (operatingTasks) and (unlockedTasks) sets, and moved to the set (finishedTasks) depicted by lines from 30 to 34. Any task waiting in (primaryTasks) that requires $t_a$to be finished will accordingly be moved to (unlockedTasks) depicted by the lines from 35 to 41. Each loop in this simulator represents a day, and it will last till the summation of (finishedTasks) reaches the number of project tasks m depicted in line 15. The last value of the simulator loops stored in $xTime$ will represents the estimated project time span depicted by line 16.

It is noteworthy that the fitness function can combine both the objective and the constraints while searching for an optimized solution. Each fitness function for each team allocation method combines in addition to the fitness functions listed above three constraints. These constraints are developed to ensure that each solution is close as possible to a feasible one. The first constraint examines whether the solution met the expectation of skills or not. The second one examines whether the solution meets the expectation of distributing all the teams or resources to the project tasks i.e. at least one resource or team is assigned to each task, and each resource or team should be at least assigned to one task. Moreover, each task or team should have no more than 12 resources assigned to it. These constraints are discussed in Section 5.2 for each method. The precedence relationship constraint on the other hand has been relaxed and the dependency constraint violation is repaired as by the fitness function described for STQS method in Figure 29 by line 16 and 24 to reorganize the task orders, and for the remaining methods depicted in Figure 30 by allowing only the tasks that have no dependency or those their predecessor(s) are finished using the four sets.

5.4 Experiment Settings and Results

The experiments performed on the four team allocation methods follow the systematic comparison process described by Chapter 3 to compare between these methods. This process starts with organizing the approaches subject to comparison according to the complexity classes they are suitable to deal with described in Section 3.3 of the dataset chapter. As the implementation of the team allocation methods is capable to handle the inputs of dependency, skills, and productivity, all these methods are suitable to all four classes, which implies that they are capable to handle all the five dataset complexity levels presented in Section 3.4. The second step of the benchmarking and comparison process is to run each approach multiple times using

the conforming dataset level to each class that the approach is capable to perform. For each run, the process suggests recording the optimal project time, and computation time metrics values. Then according to the results of each approach for each level the comparison measures should be compiled to demonstrate the efficiency, accuracy, and performance of each approach.

The results by following this process and using all the datasets are obtained using the Matlab R2016 supported by Matlab Global Optimization Toolbox. The system used for the experiments combine Intel Core m5 (1.51 Ghz) CPU with 8GB memory. Each optimized team allocation method was executed 30 times to allow determination of mean and deviation values. The GA settings for these experiments are as follow:

1. Population size: 10
2. Generation:40
3. Crossover fraction 0.7
4. Mutation probability 0.8

The motivation for these parameter settings to be very small, such as the population size, is that we need to challenge the allocation methods to the limit by which the fastest accurate results can be obtained. These challenging settings, as the intention of this chapter, can provide a clearer view and detailed information for future research of which allocation method can be used for a particular complexity level corresponds to the project time optimization problem they aim to solve.

5.4.1 Results:

The results in this section are organized according to the dataset levels discussed in Section 3.4.2. For each level, a table is presented. This table provides the results obtained for each allocation method according to the metrics and measures presented in Section 3.5. For a quick reminder, the team allocation methods are Static Teams with Queue Simulator (STQS), Static Teams with Time Simulator (STTS), Dynamic Teams with Binary Participation (DTBP), and Dynamic Teams with Participation Rate (DTPR). It is noteworthy that the hit rate measure was not used in the experiments carried out for this chapter as all the methods were able to provide a feasible solution on each run.

Level One

The attributes representing the allocation problem by the first complexity level dataset are the estimated effort of each task, and the number of resources available to perform them. Productivity

of resources in this level is set to be one. The optimal solution of project time for this dataset is 80.33 Days. The results of the experiments performed on the four team allocation methods for this level are presented in the following Figure 31 using Boxplot diagram.

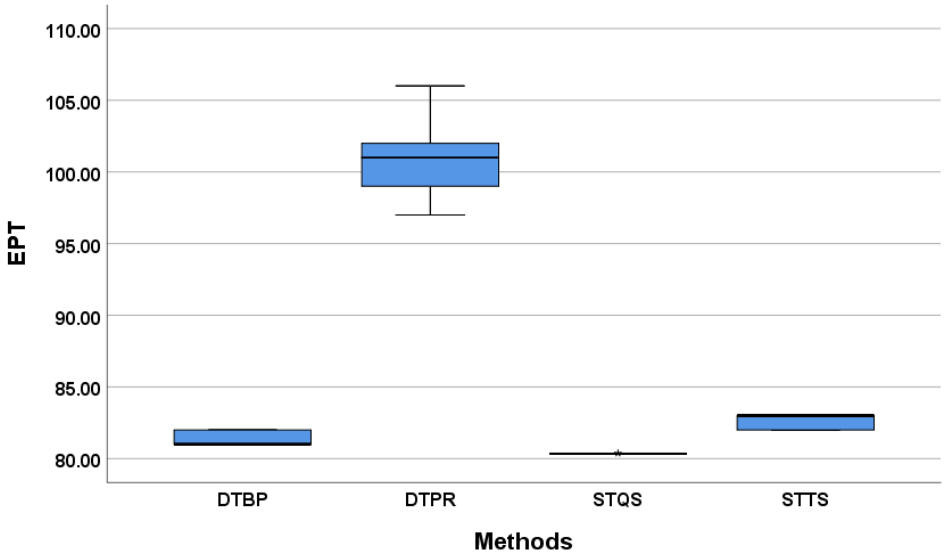

Figure 31: Team Methods Evaluation Boxplot for Level One

From Figure 31, we can see that DTBP, STQS, and STTS have produced a near optimal solutions for this complexity level. However, it can also be seen that STQS has the least variation and EPT value among the others. Accordingly STQS is the best performance amongst the other methods. On the other hand, DTPR has provided overestimates that are very high than the others, and can be counted as the worst performance for this particular level. To support this observation, detailed information of the experiment's results about the EPT and CT values are required. The detailed experiment's results using level one dataset on each method are depicted in the following Table 30.

Table 30: Results of Team Allocation Methods for Level One Complexity

	EPT	***CT***	***CT Score***	***MAAPE***	***Accuracy***
DTPR	101.06	176.47	47.78	0.252	74.75
STTS	82.70	337.9	0	0.029	97.05
DTBP	81.43	96.42	71.47	0.014	98.63
STQS	80.33	0.72	99.79	0.001	99.99

In Table 30, results of five measures are exposed to provide different perspectives about the solution quality for each method. The first measure of ETP can provide us of which method the optimal solution can be obtained. For instance, the queueing system allocation method (STQS)

can be seen by this particular measure outperforms the others with 80.33 Days of fitness function value. Given the nature of the fitness function for the remaining methods as they provide solution by simulating the daily work of resources, DTBP and STTS method can also be seen performs similarly to STQS one.

Moreover, both MAAPE and Accuracy measures clearly show STQS effectiveness in providing accurate solution with 99.99% accuracy, where DTBP method slightly behind STQS with 98.62%. From another perspective with 74.75% accuracy, we can conclude that DTPR method for this particular level is not suitable to compete with the others.

On the other hand, CT of STQS shows how fast this method in providing results with approximately 0.72 second of computation time. This has also been demonstrated by CT Score measure that show very high score of STQS method with 99.79 score point among the other methods. According to the MAAPE, CT, and CT scores, STQS method outperforms the others for this complexity level. To support this claim and capture whether there is any method that performs similar to (STQS), a paired T-Test was performed. The results of this test are presented in the following Table 31.

Table 31: Team Methods Evaluation Paired T-Test for Level One

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	STQS - STTS	-2.36528	0.46831	0.08550	-2.54015	-2.19041	-27.664	29	0.000
Pair 2	STQS - DTBP	-1.09861	0.50530	0.09225	-1.28729	-0.90993	-11.909	29	0.000
Pair 3	STQS - DTPR	-20.73194	2.02958	0.37055	-21.48980	-19.97409	-55.949	29	0.000

From Table 31, we can see that the difference in mean for each pair of STQS against the others has a 2-tailed significance less than 0.001. From these results, we have found enough evidence to suggest that the difference between the two scores for each pair is statistically significant, and reject the null hypothesis of having all the methods preform similarly.

Level Two

Level two holds three allocation problem attributes, which are the number of available resources, the estimated effort of each task, and task dependencies. Productivity of resources is also assumed to be the same of (1) for all. The optimal solution of project time for this dataset level is 80.33 Days too. The results of the experiments performed on the four team allocation methods for this level are presented in the following Figure 32 using Boxplot diagram.

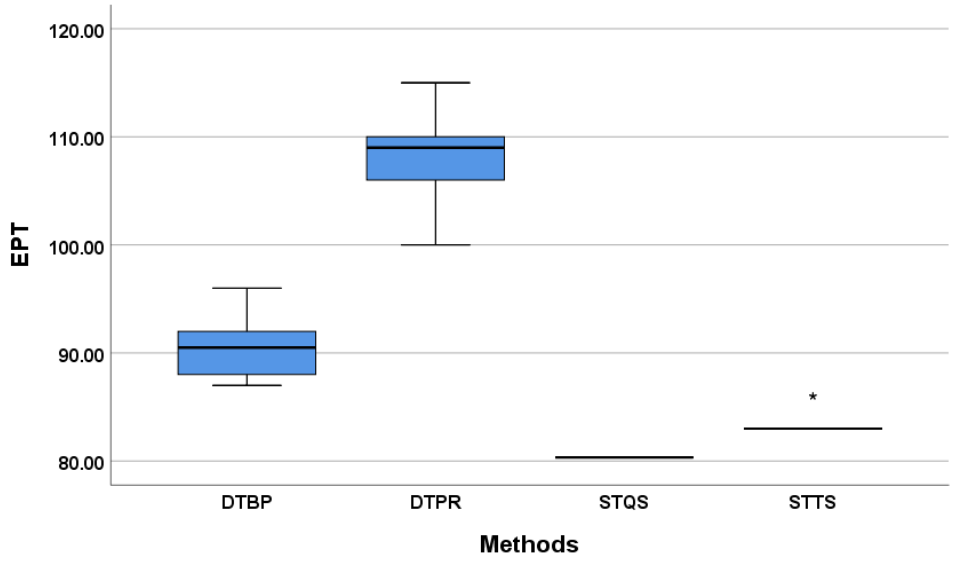

Figure 32: Team Methods Evaluation Boxplot for Level Two

From Figure 32, we can see that DTPR method is still having the worst performance and outcome's accuracy among the other methods. DTBP method, on the other hand, has slight regressed with mean EPT around 90 Days, and be the worst among the other methods. At the same time, STQS and STTS methods have both provided good quality solutions, however, STQS again has provided the least EPT values. Accordingly, this method can be seen as the best among the other methods. To support this observation, the following Table 32 provides with detailed information the experiment's results of EPT, CT, and accuracy measures by using level two dataset on each method.

Table 32: Results of Team Allocation Methods for Level Two Complexity

	EPT	*CT*	*CT Score*	*MAAPE*	*Accuracy*
DTPR	108.63	35.35	6.7	0.338	66.17
DTBP	90.63	25.62	32.42	0.127	87.25
STTS	83.20	37.91	0	0.035	96.43
STQS	80.33	1.08	97.14	0.001	99.99

From Table 32, again it is obvious that STQS outperforms all the other methods. ETP measure shows that this method has the least estimated project time over the experiment runs. In addition, the accuracy of this method depicted by both MAAPE and Accuracy measures is very high with 99.99%.

On the other hand, DTPR again can be recognized as the worst among the methods in providing good quality solutions within reasonable amount of time. However, this method is widely used by

many search-based approaches, and it might be not the best method to be adopt unless it is mimicking a real-world problem.

From these results, it can be seen that the second-best method unlike the previous level can be the STTS. This conclusion is made upon the facts shown by the measures related to project time of EPT, and Accuracy with 83 Days, and 96.42% respectively. However, CT results in this particular level shows that STQS is again outperforming the rest. The second-best method in terms of CT for this allocation problem can be seen for the DTBP one. To support the claim of STQS outperforms the others for this level and to capture whether there is any method that performs similar to (STQS), a paired T-Test was performed. The results of this test are presented in the following Table 33.

Table 33: Team Methods Evaluation Paired T-Test for Level Two

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	STQS - STTS	-2.86667	0.76112	0.13896	-3.15088	-2.58246	-20.629	29	0.000
Pair 2	STQS - DTBP	-10.30000	2.31164	0.42205	-11.16318	-9.43682	-24.405	29	0.000
Pair 3	STQS - DTPR	-28.30000	3.20004	0.58424	-29.49491	-27.10509	-48.439	29	0.000

From Table 33, we can see again that the difference in mean for each pair of STQS against the others has a 2-tailed significance less than 0.001. From these results, we have found enough evidence to suggest that the difference between the two scores for each pair is statistically significant, and reject the null hypothesis of having all the methods preform similarly.

Level Three

The allocation problem information held by the level three dataset are the number of available resources, the estimated effort of each task, as well as the skill(s) that each task requires, and each resource possesses. Productivity of resources in this level is set to be either 1 or 0.1 for each skill the resource possesses. The optimal solution of project time for this dataset level is 104 Days. The results of the experiments performed on the four team allocation methods for this level are presented in the following Figure 33 using Boxplot diagram.

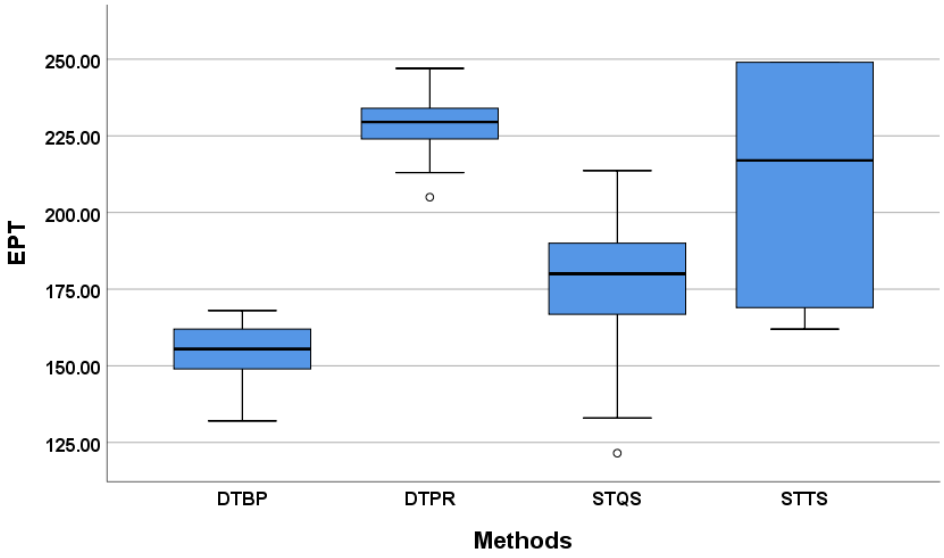

Figure 33: Team Methods Evaluation Boxplot for Level Three

From Figure 33 , we can see that DTPR method is still having the worst performance and outcome's accuracy among the other methods, and can be counted as the worst method among the others for this level too. Concurrently, STQS and STTS have slight regressed from being the best methods for level two, and it can be seen that STTS method has provided solution quality almost similar to DTPR, leaving STQS with the second least EPT values. DTBP method, on the other hand, has this time outperformed the other methods and provided the least EPT. Accordingly, this method can be seen as the best among the other methods. To support this observation, the following Table 34 provides with detailed information the experiment's results of EPT, CT, and accuracy measures by using level three dataset on each method.

Table 34: Results of Team Allocation Methods for Level Three Complexity

	EPT	***CT***	***CT Score***	***MAAPE***	***Accuracy***
DTPR	228.13	415.95	50.14	0.872	12.78
STTS	209.83	834.29	0	0.758	24.15
STQS	178.56	0.71	99.91	0.608	39.18
DTBP	154.47	117.23	85.95	0.449	55.01

From Table 34, it is clear that all the methods are overestimating the fitness function value of EPT. However, the mean of EPT over the runs shows that DTBP has the least average EPT with less variation over the runs among the methods. It is worth mentioning that the results of MAAPE and Accuracy measures for this particular level are significantly differing from one method to another. The accuracy using MAAPE and Accuracy shows that all the methods have very low accuracy,

however DTBP has the higher accuracy amongst them all. The results of computation time for each method, moreover, show that STQS method is the fastest in approximating EPT value among the others, however, that comes over the accuracy of its outcomes. The one that can be counted as reasonable in term of accuracy as well as computation time for this level is found to be DTBP method, and can be counted as the best method among them all. To support this claim and capture whether there is any method that performs similar to (DTBP), a paired T-Test was performed. The results of this test are presented in the following Table 35.

Table 35: Team Methods Evaluation Paired T-Test for Level Three

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	DTBP - STTS	-55.36667	38.70533	7.06659	-69.81947	-40.91386	-7.835	29	0.000
Pair 2	DTBP - STQS	-24.10030	23.78308	4.34218	-32.98105	-15.21955	-5.550	29	0.000
Pair 3	DTBP - DTPR	-73.66667	11.43899	2.08846	-77.93806	-69.39528	-35.273	29	0.000

From Table 35, we can see that the difference in mean for each pair of DTBP against the others has a 2-tailed significance less than 0.001. From these results, we have found enough evidence to suggest that the difference between the two scores for each pair is statistically significant, and reject the null hypothesis of having all the methods preform similarly.

Level Four

The allocation problem information held by level four dataset are the number of available resources, the estimated effort of each task, dependency between the tasks, as well as the skill(s) that each task requires, and each resource possesses. Productivity of resources in this level is set to be either 1 or 0.1 for each skill the resource possesses. The optimal solution of project time for this dataset level is 204.31 Days. The results of the experiments performed on the four team allocation methods for this level are presented in the following Figure 34 using Boxplot diagram.

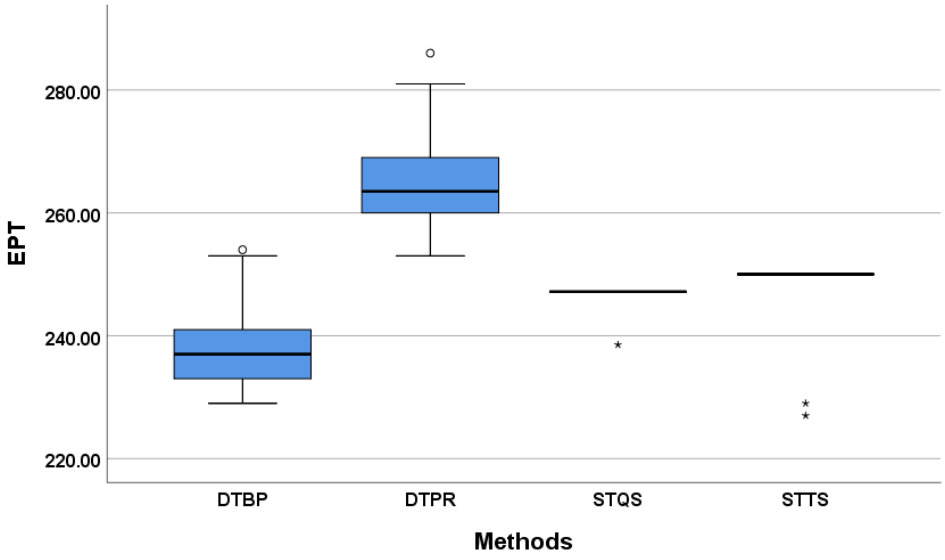

Figure 34: Team Methods Evaluation Boxplot for Level Four

From Figure 34, we can see that the same pattern of previous level results is happened, where DTPR method is still having the worst performance and can be counted as the worst method among the others, as well as STQS and STTS have provided poor solutions. What noteworthy is that both STQS and STTS have in this level provided solutions with almost no variations. DTBP method, on the other hand, has this time too outperformed the other methods and provided the least EPT. Accordingly, this method can be seen as the best among the other methods. To support this observation, the following Table 36 provides with detailed information the experiment's results of EPT, CT, and accuracy measures by using level four dataset on each method.

Table 36: Results of Team Allocation Methods for Level Four Complexity

	EPT	*CT*	*CT Score*	*MAAPE*	*Accuracy*
DTPR	266.00	82.51	15.99	0.293	70.72
STTS	248.53	98.22	0	0.213	78.70
STQS	246.89	1.17	98.81	0.205	79.45
DTBP	238.17	39.15	60.14	0.164	83.60

From Table 36, it can be seen that STQS method outperforms all the others in terms of CT, and CT score. However, when it comes to EPT and the accuracy measures, DTBP shows its effectiveness in approximating project time. DTPR on the other hand is again found to be not suitable in providing good quality solutions.

According to the accuracy measures of MAAPE and Accuracy, DTBP is the one that outperform the others with 83.59%. In addition, it can be seen the improvement in terms of CT and CT Scores

of DTBP method in providing solutions by a significant time better than STTS, and DTPR. Therefore, DTBP can be seen, for this particular complexity level, outperforming the remaining team allocation methods. To support this claim and capture whether there is any method that performs similar to (DTBP), a paired T-Test was performed. The results of this test are presented in the following Table 37.

Table 37: Team Methods Evaluation Paired T-Test for Level Four

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	DTBP - STTS	-10.36667	8.71180	1.59055	-13.61971	-7.11363	-6.518	29	0.000
Pair 2	DTBP - STQS	-8.72470	7.42406	1.35544	-11.49689	-5.95251	-6.437	29	0.000
Pair 3	DTBP - DTPR	-27.83333	8.11165	1.48098	-30.86227	-24.80439	-18.794	29	0.000

From Table 37, we can see again that the difference in mean for each pair of DTBP against the others has a 2-tailed significance less than 0.001. From these results, we have found enough evidence to suggest that the difference between the two scores for each pair is statistically significant, and reject the null hypothesis of having all the methods preform similarly.

Level Five

The allocation problem information held by this dataset level are the number of available resources, the estimated effort of each task, dependency between the tasks, as well as the skill(s) that each task requires, and each resource possesses. Productivity for each resource skill in this level can be within the range from 0.1 to 4. The optimal solution of project time for this dataset level is 112.49 Days. The results of the experiments performed on the four team allocation methods for this level are presented in the following Figure 35 using Boxplot diagram.

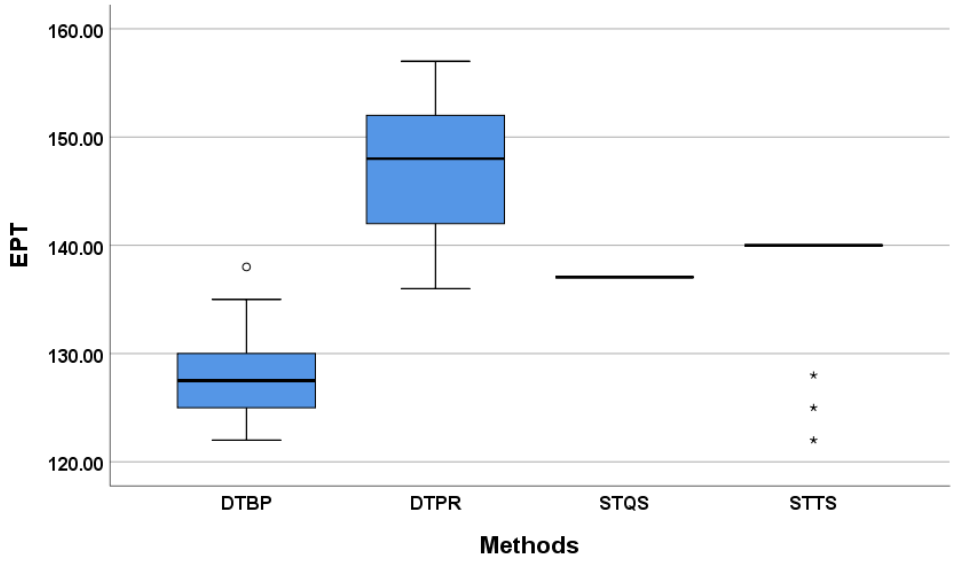

Figure 35: Team Methods Evaluation Boxplot for Level Five

Again, the same pattern of previous level results can be seen in Figure 35, where DTPR method is the worst method among the others, as well as STQS and STTS have provided poor solutions. In addition, the same results of both STQS and STTS where almost no variations between the methods outcomes over the runs. DTBP method for this level too can be seen as the best among the other methods. To support this observation, the following Table 38 provides with detailed information the experiment's results of EPT, CT, and accuracy measures by using level four dataset on each method.

Table 38: Results of Team Allocation Methods for Level Five

	EPT	*CT*	*CT Score*	*MAAPE*	*Accuracy*
DTPR	147.13	49.56	6.69	0.298	70.20
STTS	138.50	53.12	0	0.227	77.31
STQS	137.05	0.94	98.23	0.215	78.50
DTBP	128.03	51.92	2.26	0.137	86.28

From the Table 38, it can be seen by CT and CT score measures that DTBP method is again slightly better than STTS, and DTPR. However, STQS is still dominating the others in this matter. On the other hand, the measures of EPT, MAAPE, and Accuracy provide more evidence by which team allocation method the optimal or near optimal solutions of project time minimization to this particular complexity level can be obtained. The average of EPT over the runs shows that the least among these methods is DTBP. In addition, the accuracy measures show how DTBP can significantly provide more accurate solutions for this level of complexity among the others.

Therefore, DTBP can be seen again for this level of complexity as the best choice of minimizing software project time span. To support this claim and capture whether there is any method that performs similar to (DTBP), a paired T-Test was performed. The results of this test are presented in the following Table 39.

Table 39: Team Methods Evaluation Paired T-Test for Level Five

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	DTBP - STTS	-10.46667	5.89993	1.07718	-12.66974	-8.26360	-9.717	29	0.000
Pair 2	DTBP - STQS	-9.02105	3.65290	0.66693	-10.38507	-7.65704	-13.526	29	0.000
Pair 3	DTBP - DTPR	-19.10000	7.36885	1.34536	-21.85158	-16.34842	-14.197	29	0.000

From Table 39, we can see again that the difference in mean for each pair of DTBP against the others has a 2-tailed significance less than 0.001. From these results, we have found enough evidence to suggest that the difference between the two scores for each pair is statistically significant, and reject the null hypothesis of having all the methods preform similarly.

The overall findings from the performance and accuracy outcomes of each method for all the complexity levels have shown that some of the methods performed badly, others were moderate, and two methods were capable of providing good quality solutions of project time estimation. The overall weaknesses and strength of each approach that encountered by the outcomes of all the levels are presented in the following Table 40.

Table 40: Overall Findings from the Complexity Levels for each Team Allocation Method

Approach	Encountered Weaknesses or Strengths
STQS	This method can provide good quality solutions, however, only for level one and two.
STTS	This method can be rank as the second best choice for level two, however, the provided solutions for level one, three, four, and five has made it regress into the third position.
DTBP	This method has performed poorly for level one and two, however, it has dominated the level three, four, and five results with approximating project time to the best over the all methods.
DTPR	This method has provided the worst solutions over the all methods. However as it has been used by the approaches that consider project cost in addition to time objective, then it can only be adopted when intention is for a multi-objective approach.

What it can be concluded is that variability of performance and outcome's accuracy are the main characteristic that dominate the methods' performance. While STQS has outperformed the other methods for level one and two, DTBP has won on the higher levels of three, four, and five. This provide how a resource allocation method can be beneficial for one or two problems, but that does

not mean it could be beneficial for all the possible problems that a PM might encountered while allocating his/her resources. According to the overall results, for any problem that holds information corresponding to level one and two, STQS can be the best choice for approximating the project time for it. On the other hand, if the information available to the PM corresponds to the higher levels, then DTBP can be the best choice for approximating the project time for it.

5.5 Conclusion

"The only benchmark capable of combining all evaluation criteria into a decision is the decision maker himself." (p. 97) [126]. However, imitating the decision maker selection criteria can be the basic achievement towards a decision support system that can facilitate his/her work. One of the duties that a project manager, as a decision maker, has to perform is to identify and select the best team allocation alternative that minimizes his/her project time. Therefore, it is important to establish some work towards understanding and identifying the decision maker selection criteria for team allocation alternatives.

In this chapter, we have identified four main team allocation methods used by SSSP approaches. A comparison between these methods combined with GA for project time minimization is performed. The benchmarking process described in Chapter 3 was adopted in this comparison encompassing five level of complexity of dataset. The comparison shows to which complexity level the optimal or near optimal solution can be achieved by which of the team allocation methods.

The dynamic team allocation method with participation rate for resource assignment named DTPR can be seen as the worst among the allocation methods. This method has been identified and used by [14, 18] approaches, and as these methods do not employ normalization according to simultaneous tasks as in [15], these approaches have constrained the amount of overtime work for each resource to overcome the case where solution(s) combine assignment of all resources to all tasks with 100% participation. Despite the use of overtime work constraint in both [14, 18] approaches, the one that has been implemented for this chapter work is the one provided by [15]. For this case the reader can refer to [15], which in their study have used this normalization, and compare it to the one in [14]. Their comparison provides significant evidence that the one in [14] is overestimates project time, and using the normalization they suggest can provide better solutions.

On the other hand, the team allocation method that encompass queuing system named STQS has shown its effectiveness in providing optimal or near optimal solutions for the first two complexity levels. However, the accuracy measures show that the dynamic team with binary selection

representation named DTBP is outperforming the other methods by providing better solutions especially when it comes to handle complexity of skills, and productivity. The results by using the five level datasets show with clear evidence that the software project time requires consideration of the representation of team formation by the mean of the distribution of resources. This has also been demonstrated by the results of dynamic formation of teams in DTBP method, which has provided very close results to the optimized (optimal) solutions provided in the benchmarking datasets especially for levels 4 and 5 in Chapter 3 that no other alternative allocation method used was capable to provide. So, it is important to explore whether the DTBP method has some background in software industry practices for team formation and allocation. In this sense, a new study presented in [127] shows how and why software engineers move from one team to another. In their study, the main reason identified is the motivation to gain new knowledge in different specializations. However, this has motivated us to investigate the current practice not from the resources' perspective, but from the project managers' one. This investigation is carried out by the work presented in the next Chapter 6.

The main contributions in this chapter can be accordingly organized as follow. This study demonstrates how the allocation of resources in software projects with consideration to project time minimization can be formalized and performed by different team allocation methods, which have been addressed in different SSSP approaches. This study also provides information on the performance and accuracy of the identified allocation methods, which shows their performance against five scenarios.

Our intention for future work is to use the overall findings and results towards development of a management tool that can systematically define the best team allocation, which can minimize project time according to the level of information the manager can provide about his/her project and its available resources. A further intention will be focused on extending the work to include learning effects on productivity of resources. These effects can be gained by doing tasks that are related to the resource's skillset. A resource in addition, can also gain a new skill in which its associated productivity starts with 0.1, and improves over the time. We propose a learning formula similar to what has been established by [93], to represent the amount of increase in the resource skill productivity as follow:

$$Pnew = P_o * \log(time_{skill})$$

In this formula, the amount of increase on skill productivity represented as $Pnew$ is equal the old productivity of the skill P_o multiplied by the logarithmic of the time spent by the resource doing the skill over the project time span, measured in Days. We chose the logarithmic to limit the amount of improvement and to keep the improved productivity as reasonable as possible. In

addition, we aim to include the team synergy to our allocation of teams. Team synergy is important while forming the teams, and can be applied on the overall team's productivity to estimate the task time. This work can be developed based on the work of [128].

It is noteworthy to mention that the datasets used for the comparison between the allocation methods, which were provided in the benchmarking process involves a single project problem and that can be a limitation to the generality of these findings. Accordingly, expanding the datasets to include different software project problems can be an extra stage to ensure that the results are applicable to different real-world problems within the software industry. In addition, one of the weaknesses of this study is the effort estimation unit used in the dataset, which is the man-day. In case of a full estimation of effort by well-known methods as COCOMO [34], the estimation unit will certainly become as man-month, and effort then will be required by the project manager to convert the estimation from man-month to man-day.

Chapter 6 Empirical Evaluation in Industrial Settings

This chapter presents a study performed in industrial settings in which the main aim is to understand the performance of Project Managers (PM's) in finding solution to their project time problem compared to the automated SSSP approaches. In addition, this study aims to capture the difference between what the approaches propose and the current SSSP industry practice. An introduction and overview of this study, the analysis methods to use, and the research questions are presented in Section 6.1. The background for related studies on how they have performed and gained their findings is presented in Section 6.2. The methodology carried out to explore and answer our study questions is presented in Section 6.3. Section 6.4 presents the demographic information of our study subjects and the findings from their performance on solving the dataset scenarios. The conclusion of this study is presented in Section 6.5.

6.1 Introduction

The software industry is faced with a limited number of techniques and tools that can be used by PMs to support their project management activities such as Gantt chart[50], and PERT[46]. These techniques can provide graphical representation and time estimation support to PM, however they lack right decision support elements for the hardest task carried out by the PM that of resource allocation, and project staffing and scheduling. The first step towards this decision support element is the understanding of the software project properties and their relation between each other as well as the current industrial practice on resource allocation and project scheduling.

Many studies and experiments have been performed to understand and infer the relationships between software project properties as in [9, 129, 130]. However, the approaches and methodologies that software development organizations use differ from one organization to another [130], which makes it hard to bring a single SSSP optimization approach into practice. This problem can explain the reason behind the amount of work that has been done by many

researchers to approach the software project management problems. Therefore, the suitability is the aspect that a SSSP approach should focus on and implement so that it can be used by as many users as possible.

To explore the suitability of the SSSP approaches proposed for optimizing software project resource allocation, experienced PMs from software industry are the key for validating these approaches and providing to some extent their best practices and opinion on how software project management should be tackled for different management objectives. This can be achieved by first validating the SSSP approaches' inputs, the benchmarking dataset, and the associated complexity levels by the representative subjects from the industrial settings.

To this end, analysing the data need to be pragmatically studied and introduced with careful assumptions. Accordingly, it is important to know which of the data analysis methods can be beneficial for studies that relate to computer science. In literature, studies related to computing sciences have advocated hermeneutic as a valid approach to infer the phenomenon results [131]. Therefore, the way of solving different scenarios of software project complexity, and the main aspects that a PM needs to consider while performing the resource allocation, will be under investigation throughout a hermeneutic method. This method can allow to thoroughly create the overall structure for optimizing software project time and any other considerable software project aspects, parameters, and objectives.

Accordingly, the research questions that this study is aiming to answer are as follow:

1. Can a project manager solve the problem presented in the dataset levels accurately and fast?
2. Does an experienced PM perform better than an automated SSSP approach?
3. Which of the dataset levels suits the complexity of the industrial software project planning and scheduling problem?
4. Does experience play a key role in knowing the best solution for a project manager?
5. What criteria and properties does a PM look at to solve each complexity level?
6. What are the management objective(s) that the PM need to be included within his/her problem definition?

To answer these questions, it is important first to explore whether the answers can be found within the literature, but, if there is no answer for any question, then which method is best to use, and how it can be used to answer them. The following section provides background on the methodologies that have been used by different empirical evaluation and validation studies on software project management and SSSP approaches.

6.2 Background

The optimization of the SSSP problem can be classified under explanatory research, by which more understanding of the current industrial practices for SSSP is required. This can be done by building the knowledge first throughout a systematic literature review of the published papers that provide empirical evidences especially on two aspects. The first one is about the features and information that software development projects offer. The second one is about how experienced project managers practice, approach, and suggest better solution for real-world project time estimation problems. This part has already been established by [24] with a systematic literature review of all the approaches that optimize for different software project objectives. Part of these approaches presented in [24] have adopted empirical evaluation of their proposals and provided evidence on how the industry perform solutions to SSSP problem compared to what they have proposed.

Only four studies are found by [24] that have performed empirical experiments and evaluation. These studies have employed a representative sample of PMs from the industry or Information Technology (IT) students to test how they perform and provide solution to a predefined SSSP problem scenario compared to their optimized solutions. Qualitative analysis was found by [24] that mainly used by these studies to conclude of which solution between manager's intuition and the optimized approach outperforms the other. The criteria used by [24] to investigate these studies are the number of subjects, their experience, number of sessions used for the experiments, duration of each session, and the project attributes and objective.

The first study presented by [24] is the one in [93]. This study used only two senior project managers to validate their approach in a single session. The experiment session in [93] described by [24] as to capture the way of assigning the resource to software project tasks with consideration of skills within three hours limit. Another study presented by [24] is the one in [16]. This study used 16 graduate students, each of which is asked to perform an allocation that can provide a cheapest team and least schedule time for the software project scenario provided to them. The study presented in [16] used four sessions to cover all the participants depending on their availability. The third study presented by [24] is the one in [132]. It has three project managers as study subjects, however, nothing is mentioned about their experience. Their study is performed by a single session that lasted for four hours. They asked their subjects to perform three different allocations with different team sizes that maintain high team productivity and low cost. The fourth study presented by [24] is the one in [133]. This study used three managers too to validate their

proposed approach. They made individual meetings with each PM, and each meeting had four hours' time slot.

Four main findings related to these studies were listed in [24]. The first finding is that the organizations are prone to immaturity of measurement to use, with no clear development process to follow. The data used to validate the approaches are a poor institute for real-world data. In addition, studies that under their investigation have too few representative subjects. Moreover, the empirical evaluation done by the studies mentioned above have shown that the automated SSSP approaches outperform the solution performed by experts.

Another study later than [24] presented in [23] performed an empirical evaluation to show the difference between the solutions of their proposed approach and their study subjects. They have recruited 16 project managers with around four years of experience each. They have provided those subjects with the data they used to test their automated approach and the mean of their results are then compared with the mean of the approach's outcomes. The main findings from their experiment again is that their approach outperform the solution provided by the experts.

In addition, they have performed a pilot study to qualitatively survey the real needs of a software organization. The aim of their survey is to capture the aspects that managers from industry consider while performing resource allocation and scheduling, and whether the different constraints and objectives used within their approach are suitable for project managers' needs from an automated approach. The findings from their survey are that managers consider reducing the amount of parallel work of each developer as much as possible, with an intention to minimize project time span. In addition, they found that with a high importance for project managers, is to make sure that each resource with his/her skillset fits to the task(s) that (s)he will perform.

In general, surveys and questionnaires are used in studies that have performed empirical validation of their proposals as in [93]. The responses of their subjects are qualitatively analysed to provide evidence of their solution quality. These studies however lack detail of the research and analysis methods used. These methods, how they have been used, and the results of applying them to experiment with and evaluate SSSP within industrial settings are provided in the next Section 6.3.

6.3 Methodology

To perform this study, identification of main steps, methods, and procedures was made, which all are established as a framework and protocol for the participants and the researchers in this study to follow depicted in Figure 36.

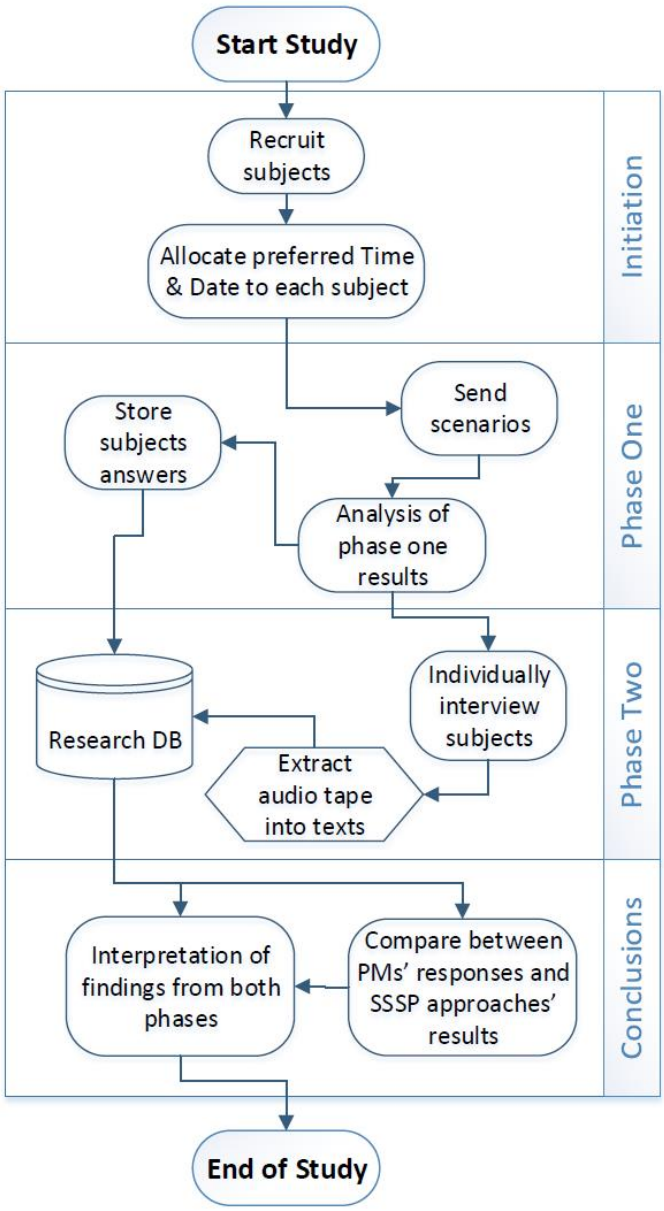

Figure 36: Methodology of the Industrial Evaluation Study

As can be seen in Figure 36, the industrial settings evaluation study starts by recruiting subjects from software organizations. At this stage, the subjects will be recruited from our industrial partners using a direct recruitment method. The main recruitment criteria is to have a PM with at least five years of experience in managing software projects, to gain by his/her cumulative experience more in-depth knowledge about the important aspects that should be covered while managing HRA in software projects and their preferable way of doing it. More demographic information about the subjects are provided in Section 6.4.2.

This study is planned into two phases depicted in Figure 36. The outcomes from both phases are quantitative and qualitative research data. The analysis of the data will be performed by employing quantitative and qualitative methods. The first phase involves quantitative data analysis methods for examining the solutions of the recruited PMs in solving different resource allocation problems depicted in Figure 36. These resource allocation problems are the first four scenarios of our datasets in Section 3.4.2. These scenarios will be sent to the subjects so they can complete them in their own time and send them back. The responses of the subjects will be then stored in our research database. This phase is detailed in Section 6.4.1.

When the subjects will be asked to provide their answers in the first phase, they will also be asked to provide a suitable date and time for the second phase. This second phase encompasses interviews that should allow to extract more information about the subjects' responses from the first phase, and to explore their demographic information. Accordingly, once the subject completes his/her answers to phase one, and his/her responses are analysed, an interview then should take a place to meet him/her, and to discuss his/her views, opinions, and knowledge about the scenarios and the whole aspects surrounding software projects –see Section 5 of the Appendix. The interview with each subject should be individually, and planned for one hour time slot if the subject's time permit. The responses of study subjects from phase two should be then extracted and stored in the research database. Based on the subjects' responses and extracted data from both phases, a comparison can be performed between their solutions and the solutions obtained by the SSSP approaches -presented in Chapter 4-, and interpretations according to the demographic information can be then made.

The overall objective of this study is to provide basic research that expand our understanding and knowledge in SSSP problem including the DM activities in this matter more than to provide a definite solution for it. For this purpose an interpretivist approach [134] is adopted combined with the interview method to collect and qualitatively analyse the data. However as the study phases combine quantitative and qualitative data, a mixed method is used to analyse the outcomes from the study subjects [115]. The data collection methods used are self-completion questions, interview, interview-structured questions, and a face-to-face or internet-conferencing meeting. For data analysis two methods are used, which are pragmatic hermeneutic, and statistical analysis.

6.4 Study Experiments

This study was performed upon the approval of the ethical clearance provided in Section 2 of the Appendix, where in addition the application, description, questions, and protocols of this study can be found too. Seven subjects were recruited for this study, and before any meetings,

interviews, and questions took place, we have sent a consent –see Section 3 of the Appendix- to each subject. The subjects have requested to remain anonymous in any output from the two phases. The only thing that we can mention about them is that five subjects are from a large financial organization that consists of in-house software development departments, one is from an international software development company, and one is from a start-up company with more experience from another large software development organization. It is worth mentioning that we had two cases were a contact person was assigned to organized the communication, meetings, and interviews with the subjects from the large financial organization.

6.4.1 Phase One: Evaluation of PMs' Performance in solving SSSP Challenges

In this phase, the study subjects are asked to answer a set of four self-completion questions. Due to time limitations and availability, the subject can chose when to complete these questions and send them back by email for analysis. This set encompasses four SSSP scenarios corresponding to the classes and complexity levels discussed and solved in Chapter 3. The subjects in this part are asked to provide their best allocation and estimated project time span according to the resulting allocation schedule, and the time that they consumed to solve each question. As the scenarios' data provided to subjects are the same as those used for the SSSP approaches evaluation, this will allow us to demonstrate whether the subjects perform the allocation in a similar way to the approaches, and to infer the factors that might play a key role in contributing to good quality results from the subjects.

Findings and Results

One of the subjects (D) did not provide answers to this phase's questions, and asked to proceed to the next one. The reason given by the subject for this matter is the absence of real factors that these scenarios did not include such as the description and details about the intended software to be developed as well as the roles that are required for a single team to perform the project activities. However, the subject claims that it can be seen within the large size companies' projects, similar to the one presented in scenario four, where tasks, dependencies, skills, and productivity are all that a PM would consider while performing the allocation. For details on the scenarios provided to the subjects the reader can refer to Section 3.4.2.

In addition, another subject (M) has only provided answers for the scenarios but could not proceed to the next phase. The problem was the time availability and implication of the different time zone, as the subject is constantly traveling. The solution provided by each subject in this study of

estimated project time span and the time consumed to solve each scenario are presented in the following Table 41.

Table 41: Study Subjects Responses

Participant Name		D	T	N	C	S	E	M
Scenario1	*Time Performance/Minutes*	NA	7	60	15	20	20	22
	Min Solution -Days	NA	88	81	111.5	111.5	111.5	111.5
	Allocation Method	NA	Dynamic teams	Dynamic teams	Rigid teams with Percentage	individuals, no simultaneous works	Two resources to each task	Dynamic teams
Scenario2	*Time Performance/Minutes*	NA	10	60	40	50	60	44
	Min Solution -Days	NA	87	82	412.5	412	442	111.5
	Allocation Method	NA	Dynamic teams	Dynamic teams	Rigid teams with Percentage	individuals, no simultaneous work	Rigid teams	Dynamic teams
Scenario3	*Time Performance/Minutes*	NA	15	75	40	30	30	28
	Min Solution -Days	NA	104	104	132	132	132	117.8
	Allocation Method	NA	Dynamic teams	Dynamic teams	Rigid teams with Percentage	Rigid teams	Rigid teams	Dynamic teams
Scenario4	*Time Performance/Minutes*	NA	17	80	60	75	70	58
	Min Solution -Days	NA	248	210	1188	478.5	442	479
	Allocation Method	NA	Dynamic teams	Dynamic teams	Rigid teams with Percentage	Rigid teams	Rigid teams	Dynamic teams

Table 41 presents the subjects' responses who participated in this phase for PM performance study. As their identity remains anonymous, letters are used to differentiate between the subjects' identity in all the associated tables and statements. This table moreover includes each subject's answers of estimated project time, the time consumption, and allocation method (s)he adopted to solve each scenario. The allocation methods that each subject adopts to solve the scenarios differ

from one to another, and some have even used different methods to each scenario. When the allocation method is “dynamic”, this means that the subject has allowed a team to change its members from one task to another. “Rigid team” on the other hand, is when the team has its members from the start of the project working together till the end without any changes to its members. “Percentage” moreover, is when a resource is working simultaneously on multiple tasks, so each task has a percentage of his/her working time dedicated for completing this task. Another type that has been used by one of the subjects is individual allocation that considers allocating only one resource to each task. Moreover, another allocation was also made by allocating two resources to each task.

The overall inferences from the results shown in Table 41 can help to understanding the subjects’ behaviour corresponding to each scenario. From Table 41, it can be seen that as the scenarios’ level increases from one to four, the subjects in general spend more time to solve that scenario than the one before, leaving scenario four taking the highest time to solve. It is also noticeable that both T and N subjects were able to provide good quality answers as their project time estimate is too close to the optimized (optimal) one. It is interesting to consider why those two subjects were able to provide such good answers. This situation shows why the work for this thesis has supplemented phase one by the second phase of interviewing subjects to gain more explanation.

In addition, it can be seen that the subjects for scenario three have close results to the optimal one as by subject C, S, and E with 132 days, and to some can be even more identical to the optimal one such as subject T and N with 104 days. This reflects the simplicity of the scenario’s attributes and how the subjects are familiar with this situation so they have responded well to this scenario. It can be seen too that the dominant allocation method adopted across the subjects’ answers is the dynamic team method.

It is noteworthy that some subjects, were not only trying to minimize project time, but they were also trying to balance the allocation of twelve resources over the whole project, even with the dependencies between these WPs. This can be seen over the solutions of subject C, S, and E. For instance, subject S have created a list, titled “age allocation”, that provides the percentage of the work load, having the WP’s effort divided by the overall project effort, over the number of resources. Subject S used this list to know how many resource (s)he should assign to each WP. By using this way of allocation while balancing the more skilled resource to the most fitted WP, it could end with the result of subject C having the least skilled and productive for a WP that has a very high estimated effort amongst the others.

Analysis of PMs and SSSP approaches' solutions

Comparing these results with the ones obtained by the nine SSSP approaches, one can see how in some cases some of the PMs have performed the resource allocation and project scheduling similar to the approaches, and in others the PMs have performed badly. For instance, if we look at the best SSSP approach like ***DiPenta01*** for level one and the best PM's result by subject N for the same level of (scenario 1), it is obvious that the subject was able to provide a good quality answer similar to the one of ***DiPenta01***. However, the subject has consumed of one-hour time to find this answer. For the same level, in addition, five subjects have provided bad solutions, such as the answer of subject M with 115.5 days of estimated project time. The answer of this subject was based on having all the tasks starting at the same time where a single resource assigned to each, and for cases of a large task size such as task 2, 3, 4, and 5, two resources are assigned. Therefore, the estimated project time defined by the subject is the maximum time length among all the tasks with 115.5 days of task 2. It is noticeable that the approach in ***Kang01*** has provided exactly the same estimate as those PMs.

For level two, we can see that subject E has performed badly too providing 442 days of estimated project time. The subject has created the resources plan with a static view of resource allocation regardless the precedence relation between the project tasks that can allow for a dynamic allocation to be used. For instance, the subject's plan has misused the resources who are assigned to proceed task(s) by being idle till the precede ones are finished. The subject has assigned three resources for task 2, two resources for task 3, two resources for task 5, and a single resource for each of the other tasks. This can show clearly how subject E had a static view of the resource planning, where resources are distributed to tasks without any care of schedule. It is noteworthy that none of the SSSP approaches described in Chapter 4 has provided similar to this bad estimate. The worst estimate provided for this level of complexity is by ***Minku01*** of 109.17 days, which is clearly show how a SSSP approach can help with resource planning and project time estimation, as PMs with years of experience are struggling to provide similar estimate.

For level three, the PMs were able to provide much better estimate than the SSSP approaches, as this level requires only a direct matching of resources' skillset to tasks. The SSSP approaches, on the other hand, have provided a very fast estimate to project time. However, these estimates are very poor if we compare them to the PMs and optimal ones. So, for this level we can say that the PMs can outperform the approaches. Nonetheless, the worst project time estimate can be seen by three subjects' (C, S, and E) solutions of 132 days. This poor estimate can be explained by the solution that subject S has provided. This subject has created his/her schedule by the assumption

of making the tasks work in parallel and assigning the competent resources who are possessing the required skill(s) to each task. As the project has four skillset categories, the tasks are divided by these categories, where each particular skillset is required by two tasks. On the other hand, only three resources possess the required skill(s) for each type. The subject's decision was then on assigning two competent resources to the task that is larger in size among the set, and the third resource to the smaller one. Noteworthy that this has made the subject leaves task 4, which is smaller in size compared to the one that requires the same skillset but larger than many of the other tasks, assigned to a single competent resource. This has accordingly led to the longest time estimate among the parallelised tasks of 132 days.

For level four, both SSSP approaches and PMs have provided similar estimates, where the performance can be the key subject that shows the difference between the approaches and PMs. In this case, it is obvious that the SSSP approaches are able to provide much faster estimate than the PMs. However, if we look at the best estimate among the PMs and approaches, we will find that one of the PMs (Subject N) has provided much better estimates than all the other approaches and PMs with 210 days of project time. On the other hand, a very bad estimate among the PMs' can be seen by subject C, with 1188 days of project time. This estimate is based on distributing the resources with a percentage for participation to project tasks. The estimate by this subject had in general two resources assigned to each task, except task 5 with one resource. In addition, the resource who is possessing the required skill(s) was assigned to task 3 with 50% participation and the other who don't possess these skill(s) was assigned to this task with 100% participation. Moreover, the resources assigned to task 2 are possessing the required skills, however, they have been assigned with 50% participation to each. For task 4, one of the resources assigned is possessing the required skill(s) and the other is not, and both were assigned with 50% participation. For task 6, the same theme of resource allocation to task 3 and 4 is used with two resources that one is possessing the required skill(s) and the other is not, but both are participating in this task with 100%. For task 5, a single resource is assigned to this task however, with 50% participation. The time estimate for tasks 3,2,4,6, and 5 are 300, 223, 240, 45, and 380 days respectively. While these tasks forms the critical path of the project schedule, these estimates have all together formed the project time estimate of 1188 days.

This survey has shown how hard the SSSP problem is for a PM to consider its all parameter while focusing on the optimal project time target. The main theme that these PMs have used is balancing the amount of resources assigned to tasks without consideration of the idle time that could this assignment cause on the overall project time.

To express how an average PM would probably perform in solving the scenarios is to average the subjects' solutions. The following Table 42 summarizes the average of the subjects' responses compared to the optimized (optimal) solution provided for each scenario in Section 3.4.2. Note that the average of the responses is calculated based on six subjects as the seventh one has no response recorded for this phase.

Table 42: Evaluation of Study Subjects' Performance

PM challenges	*Attributes*	*AVG of Subjects solutions*	*Optimal Solution*
Scenario1	*Time Performance/Minutes*	24.17	80.33
	Min Solution -Days	102.5	
Scenario2	*Time Performance/Minutes*	44	80.33
	Min Solution -Days	257.83	
Scenario3	*Time Performance/Minutes*	36.3	104
	Min Solution -Days	120.3	
Scenario4	*Time Performance/Minutes*	60	204.31
	Min Solution -Days	507.58	

For each scenario, two main attributes are depicted in Table 42. The first attribute of "time performance" presents the average of time consumed by the subjects to solve the scenario represented in minutes. The second attribute of "min solution" is the average of estimated time span of the corresponding project scenario, represented in terms of days. It can be seen from Table 42 that scenario one and three on average are simpler for a PM to solve than when dependencies and/or skills and productivity, represented by scenario two and four, have to be taken into consideration. From Table 42, it can be seen that the subjects were able to provide answers for scenario four, which is a simple project in size, within 60 minutes on average, however, their average of estimated time span is the double of the optimal one with estimated 507.58 days to complete the project. That shows how hard the SSSP problem is. A question has arisen as to why for those two scenarios the subjects were able to provide good answers. To explore more about the subjects' knowledge and background, and their demographic information the following Section 6.4.2 discusses these aspects.

6.4.2 Phase Two: Follow-up Interview for Qualitative Study

This phase is performed by interviewing the subjects with at least one hour time slot for each. The interview is carried out according to the subject's meeting preferences either through face-to-face or internet-conferencing meeting. Three subjects were unable to have face-to-face meeting due to their location that is far from University of East Anglia (UEA).

The interviews combined follow-up questions within the interview-structure, and opened new room for discussion. The interview questions can be found in Section 5 of the Appendix. The results of these questions are discussed according to the questions' categories, which are divided into seven. These categories are:

- The organizational size level that the subject represents.
- The subject's project management experience.
- The project attributes that the subject thinks are important.
- The allocation method that represents what the subject practices.
- The considerations that the subject thinks a PM has to think about while forming a team.
- How the subject do his/her project scheduling. and
- The objectives that the subject thinks it represent the management goal(s).

For more details on these categories and their detailed questions the reader can refer to Section 5 of the Appendix. Note that subject (M) did not complete with us in this phase, and we were unable to explore any of his/her demographic information in this phase. In addition, subject T, N, C, S, and E are all from the same organization, but from different geographic branches' locations. The outcomes from interviewing the six subjects for the organizational level and project management experience categories are presented in the following Table 43.

Table 43: Responses of Study Subjects for Organization Level and Experience Interview Categories

	Subjects						
Study Aspects	**D**	**T**	**N**	**C**	**S**	**E**
Organization size level	*Medium size*	*Large financial organization*	*Large financial organization*	*Large financial organization*	*Large financial organization*	*Large financial organization*
Experience	**Years of PM experience**	*6*	*26*	*30*	*17*	*10*	*17*
	Development methodology	*Agile*	*Agile and waterfall*	*Waterfall*	*Agile and waterfall*	*Agile and waterfall*	*Agile and waterfall*
	Project Sizes	*Medium size team*	*Large and multiple teams*	*Large and multiple teams*	*Large and multiple teams*	*Large and multiple teams*	*Large and multiple teams*

From Table 43, it can be seen that the main participants were from large size organizations. However, their years of experience vary from one to another. The least experienced in management can be seen in the table as subject D, whereas subjects T and N are the most experienced amongst them all with four years difference between them. It can be seen too that the subjects who represents large organization with large project and teams' size are combining agile and waterfall models in their projects. This confirms the observation reported in [130] that the

large organizations are in favouring of using hybrid methods, which combines different development methodologies as waterfall with other(s), over the agile approach.

The responses of subjects for the aspects of project attributes and resource allocation are depicted in the following Table 44.

Table 44: Responses of Study Subjects for Project and Resource Allocation Attributes Interview Categories

		Subjects					
Study Aspects		*D*	*T*	*N*	*C*	*S*	*E*
Project Attributes	*Productivity*	Commitment ratio	Analogy	Analogy, learning and synergy with others	Story point, and personality between the team	Story point, and personality between the team	Story point, and personality between the team
	Scenario relevant to the organization	Four	Four	Four	Two	Two	Two
	Important Attributes	As presented in scenario four	As presented in scenario four	As presented in scenario four	As presented in scenario two	As presented in scenario two	As presented in scenario two
Project Resource Allocation	*Resource Allocation Method*	*Leave each to pick from a list of tasks*	*Create different permutations of agile team according to their velocity*	*Dynamically change resources from one task to another as the need for skills and proficiency*	*Form a single team that works coherently for a single target*	*Form a single team that works coherently for a single target*	*Form a single team that works coherently for a single target*
	Nature of team assignment	*Individuals to project*	*Dynamic teams*	*Dynamic teams*	*Rigid teams*	*Rigid teams*	*Rigid teams*
	What do you think of dynamic assignment	*This is how it works in reality*	*This is how it works in reality*	*This is how it works in reality*	*This is how it works in reality*	*This is how it works in reality*	*This is how it works in reality*

From Table 44, the presence of resource productivity can be seen among the subjects' interpretation. However, each has represented productivity according to his/her practice. For example, subject N has described productivity as the analogy of a resource compared to his/her colleagues with respect to learning speed and synergy with the team members. Others have almost the same concept as they represent productivity by the speed of developing story points, and in relation to other team members, as with subjects C, S, and E.

In addition, it can be seen from the Table 44 that half of the subjects claim the existence only of scenario two, which only consider dependency between project tasks. It is worth mentioning that they all understand that the resources they have in their organization or company are sharing similarity in terms of skills and productivities, as the HR department applies standards and quality check. However, the other half of the subjects support the existence of scenario four as a reality of

complexity level they face within their projects, and they do believe that the resources differ in their skills and productivity. What is also important to mention, that all the subjects do not use productivity as a factor while they staffing and scheduling their projects.

The method adopted by each subject differs from one to another for their project resource allocation. Subject D leaves the resource to decide their tasks. For subject T, (s)he allocates the resources to teams after creating different permutations so (s)he can decide which one is better based on the team velocity. Subject N allocates his/her resources to teams according to their skills, but when the expertise is required for another team working on another task, they can change teams' members. A consensus can be seen with subjects C, S, and E to allocate resources to a single team where the personality factors plays the core role to create a coherent team, and to avoid any conflicts between the members. The nature of their teams can be seen in three different types. A single team that each member works on his/her own task(s) for the same project as for subject D. Another type is when the resources assigned to a single team that will perform having the same members without any changes from the start of the project till the end. The last type is when the teams can change their members from one task to another based on the expertise needed for new tasks. This type of team formation however, has a very high chance to occur in software projects as all the subjects reported that in the last question of the resource allocation category.

In addition to the previous categories, team consideration, and project scheduling aspects were also subjects for discussion with the study subjects. The responses from each subject towards these aspects are presented in the following Table 45.

Table 45: Responses of Study Subjects for Team and Scheduling Interview Categories

		Subjects					
Study Aspects		*D*	*T*	*N*	*C*	*S*	*E*
Team Consideration	*Assignment Criteria*	Cross-functional team. Not sharing same expertise	Behaviour, performance, and technical skills	Cross-functional and technical skills	Cross-functional team. No sharing between multiple tasks. Scrum master provide us with the very skilled to project to make the development.	Cross-functional team. No sharing between multiple tasks. Scrum master provide us with the very skilled to project to make the development.	Cross-functional team. No sharing between multiple tasks. Scrum master provide us with the very skilled to project to make the development.
	Teams' Skills Nature	roles	Roles, and technical skills	Roles, and technical skills	Roles, and technical skills	Roles, and technical skills	Roles, and technical skills
Project Scheduling Allocation	*How do you Recognize Dependency*	*According to money cost of each task*	*Spikes of story points*	*Similar to the scenarios where also requirements, legislation, and other outside aspect we consider*	*dependencies with different respects to the internal and external aspects*	*dependencies with different respects to the internal and external aspects*	*dependencies with different respects to the internal and external aspects*
	Single or Multi Project(s)	*The problem still the same*	*Single as dependency will make it similar*	*Multi-project, with consideration of availability*	*Multi-projects*	*The same as how it works for a single project*	*single*
	What do you think of Dependency	*Not always the case*	*We try to avoid as much as possible*	*Resource availability*	*Internal and external aspect such as resource availability with respect to other projects*	*Internal and external aspect such as The percentage of your resource availability with respect to other projects*	*Internal and external aspect of outsourced components, legislation, and risk mitigation*

Table 45 above shows the aspects discussed with the subjects regarding how they team up their resources and what criteria they use to do so. For this matter the subjects have demonstrated their team formation criteria by showing what skills they consider. Broadly speaking, all the subjects consider cross-functional teams that combine different roles supported by technical skills as the development needs specific languages and technologies to be used. In addition, Table 45 shows the outcomes from the discussion with the subjects regarding project scheduling. The first question in this category is about how the subjects recognize dependency between project tasks. The subjects have different criteria in this regard. For instance, subject D do the schedule for the

low cost/high revenue task to be done first, and then iteratively complete the whole product. Others see the real dependency that connects one development task according to stories or requirements to another as precedence relationship between the tasks. This question of schedules is planned to be followed by whether the subjects think dependency should also include multi-project environment while scheduling and staffing a software project. Two subjects were explicitly requiring the consideration of multi-project environment explained by the availability of resources as they can be allocated to another project(s). Other subjects do not see the difference as the schedule should include dependencies between the projects and their interdependent tasks. What is noteworthy to mention is that subject T avoids inter-dependent projects as much as possible due to the rework that can potentially occur during the development.

The objective(s) that the subjects consider while they are staffing and scheduling their projects are presented in the following Table 46.

Table 46: Responses of Study Subjects for Management Objectives Interview Categories

		Subjects					
Study Aspects		D	T	N	C	S	E
Project Management Objectives	Is project time the ultimate objective	yes	yes	No	yes	yes	No
	What other objectives for your projects	None	Customer satisfaction and Quality	Cost, and time	Quality	None	Cost, time, and quality
	Should cost be considered	No	Yes	Yes	yes	No	Yes
	How do you do costing	Based on time	Based on time	Based on time	Based on time	Based on time	Based on time

Table 46 presents the answers of the subjects regarding the objectives they consider for their software project management. This category has four questions where the first addresses whether the subjects agree with time being the ultimate objective. The second question addresses additional objectives that the subjects think are also important to consider. The third question captures whether project cost in particular should be considered if the subject did not address it in his/her answer to the second question. The fourth question addresses the method that the subject use to estimate project cost.

Four out of six subjects consider minimizing project time as the ultimate objective, two of which do not include any other management objective to their projects. The reason behind their opinion

is that project contracts, either for internal within the organization, or external for a customer, place great emphasis on the deployment and delivery date of software projects, which limit the project time more than any other management goals. This can be summarized as stated by subject S “To finish the project on time and as planned is the most important thing” and as subject D stated too that “The cost is time”.

It can be seen from Table 46 that some of the subjects have included more objectives than project time minimization in their answers. Four out of six subjects see that project cost minimization should be considered within the software project management decision. However, only subject N and E have explicitly mentioned project cost in their answer. When the subjects are asked by the researcher on how they do cost estimation, a consensus can be seen among them as they all have stated that cost should be calculated based on the amount of time spent by the resources to develop the software. This however can be captured by the utilization of resources that takes into account resources’ availability, in which the amount of usage of those who are experts and have their salaries higher than the others will reflect on cost with a positive relationship. For this matter subject D added that “software project’s cost cannot be affected by resource allocation, it depends on how many months and experts the development of product will take and that should include the running cost of that department“.

In addition, three out of six subjects have added maximizing product quality in addition to project time. Moreover, subject T has added customer satisfaction to the management objectives. Despite the fact that customer satisfaction is a bold one that might encompass all the other discussed objectives, these management goals are crucial to PM to maintain, however it is hard to balance between them as they are conflicting each other. To clarify how it works within an organization to provide identification to all these objectives, subject S stated that “quality and cost are something that happen behind the scenes, and are agreed before the project starts, so as a project manager I am left to maintain the project time and schedule more than any other things”.

6.5 Conclusion

The industrial settings evaluation presented in this chapter has demonstrated the complexity of managing software projects and the variability of the attributes, development methods, and resource allocation approaches adopted by the PMs. Questions that gradually emerged throughout the study helped to identify and shape the study aims. Our first aim is to define, and search the attributes, aspects, and parameters that the PM uses in staffing and scheduling software projects. The second aim is test the suitability of the dataset used in this thesis, and its levels. The third aim is to compare between the study subjects’ solutions against the same optimal (optimized) ones

used to evaluate the outcomes of SSSP approaches. The fourth aim was to search the possible trends and future directions for this research field. Meeting the above aims will help us learn about the suitability of SSSP approaches for industrial adoption.

The aspects of staffing and scheduling software projects

In meeting our first aim, we have identified by our study subjects that the task dependency, resource's skillset and productivity are important to be considered by the PM while staffing and scheduling a software project. Moreover, the resource's availability for project tasks is another aspect that has been identified by the subjects, which should be included within the SSSP problem. The management objectives that our study subjects believed are important to be considered varies from between only the time span, or the three of time, cost, and quality. Broadly speaking, the management objectives are to minimize project time and cost, and to maximize the outcomes quality. However, some of the subjects have made it clear that after the contract agreements project time is what they left with to manage corresponds with the findings in (p.33) [135].

The precedence relationship between project's tasks (task dependency) was judged by all the study subjects as one of the important parameter for staffing and scheduling software projects. Each subject however believes that (s)he has his/her own expression for this terminology. For instance, dependency has been illustrated by five subjects as waiting for other work to be delivered so the task can be started, whereas the other subject understands the dependency as a priority where the task with higher priority should be performed first. Despite these expressions it can be seen that they are all leading to the same definition of which task should be performed before the other, and that shows how a consensus across the subjects is for dependency definition.

In addition, the human resource skillset has been recognized by three subjects as an important attribute to consider while allocating the resources to a software project. It is important to mention that those subjects are programme managers by which their job combines different services, departments, and projects into a consolidated programme with many project managers to guide. Moreover, variability of a resource's productivity is also found by those subjects as a factor that they do not consider for staffing, but it is a reality that should not be ignored. Subject E stated on this matter that, "skills and productivity is never represented as in the scenarios. However, we have a performance check measure on each period of time for each employee so that we make sure that everyone is up to the standards of software development projects". Additionally, subject S stated that "we ask for a resource and the scrum master provides us with the most suited to and productive for the task(s) we need him/her for". Furthermore, subject S also added, "We prefer

good plus resource than an expert, so we can ensure cohesion between the team and no one can have his/her influence on the rest".

In addition to the previous factors, resources' availability has been identified by the study subjects for its importance. This attribute was addressed by two PMs on how it plays a critical role and has affected the resource allocation, staffing, and project scheduling in a multi-projects environment. This attribute moreover, has an influence on project time and resource's participation percentage to different tasks and projects, which can lead to negative implications on the overall project progress. Despite the importance of this factor, the aim in this thesis is to evaluate the SSSP approaches that have considered single project to optimize its time, which accordingly leads us to ignore this factor at this time.

Which of the scenarios are most likely to represents the PMs' project complexity?

In order to investigate the suitability of SSSP approaches for our second aim, we should test the suitability of their inputs. This step can be done by validating our datasets by PMs from software industry. We have met this aim by the responses from our study subjects as they judged two of the datasets to be similar to what the industry faces. This has been established by knowing which scenarios are representing their problems.

Three of the subjects have seen their project management problem demonstrated only by scenario two as they already assume that their resources are skilled, and productive ones. However, from their statements when productivity was under discussion, they all agreed that human resources in software projects do differ in their productivity and skillset they possess from one to another, and that those two attributes have to be considered while managing software projects. Therefore, the project complexity level that draws the PM's main attention is that represented by the scenario that includes multiple interdependent tasks, and resources' skill set and productivity attributes. However, a problem seems to arise as some have refused to use any approach that depends on effort estimation, nor size of software project task, and they claim that this might prevent them from using any approaches with outdated effort measures stated by subject D.

While the time equation that SSSP approaches adopted are mainly depending on the division of the amount of work over the amount of progress that can be achieved in a unit of time, this representation can easily be adjustable to different working units as the PM needs. For instance, if the main development method that the organization uses is agile, then the unit of measure can be easily adjustable to user-story, story point or even micro-services. Therefore, the measures that a PM can use with a SSSP approach can be adjustable to these kinds of units. Accordingly, there

is no need to change the foundation of these approaches since the time measurement stays the same.

How did the PMs perform in solving the scenarios?

In meeting the third aim and answering question 1, 2, 4, and 5 in Section 6.1 we can conclude the following. Study subjects who participated were PMs with 6 to 30 years of experience. It appear to be that the more experienced PM can perform better than an average one, and similar to an automated SSSP approach on the problems presented in our particular datasets. Less experienced subjects spent more and more time on solving the scenarios for less quality solutions as the complexity level increases. This shows how experience plays a critical role in finding optimal or near optimal solutions to SSSP problem.

Scenario one and three appeared to be straightforward for PMs to find their near optimal solutions. However, some of the study subjects struggled to provide a good quality solution to these scenarios. Moreover, as the complexity increases some PMs were unable to find near optimal solutions. The time performance of PMs in solving the scenarios, ranges on average for the simplest with 24 minutes, to the hardest with 60 minutes. Having in mind that the scenarios combine simple project with eight development tasks and twelve resources, this study has demonstrated how hard SSSP problem is, especially when scaled up with more tasks and resources.

In addition, it is important to capture the difference between the PMs' practices by which it is noticeably that different allocation and team formation methods were used by the study subjects and the method(s) adopted differ from one PM to another. For instance, some PMs have assigned rigid teams to project tasks where others just dynamically changed (shift) members from one team to another over the progress of project tasks and time. This practice however, can contribute to boosting the resources' productivity if the resources are able to select what they think is suitable for them as reported by [127].

Moreover, it is interestingly to observe how some of the subjects have assigned some resources with percentage to work on simultaneous tasks and teams attempting to increase the number of workers and to reduce the amount of development time on these tasks. This practice however has been addressed by many researches as in [120, 121] on how it can reduce resource's productivity and project progress, and is unlikely to produce good quality solutions. In this matter, it can be concluded as subject D stated that "it is hard for a resource to work simultaneously on different tasks together". Therefore, adopting the dynamic allocation with consideration for singularity of

assignment at a time for each resource could help especially in an environment where dependency of tasks, skillset, and productivity of resources should be considered.

The possible trends and future directions for this research

For our fourth aim, we have found that the subjects' responses varies in terms of the resource allocation method, criteria of resource selection, and project properties they consider while allocating the resources to project tasks, which worth more exploration by a future work. This can be linked to investigate why the more experienced PM, as subjects T and N, tends to provide such high quality solutions, and to imitate their choices by an optimization approach. Moreover, resource availability is an aspect that worth to investigate for the possible ways to integrate it within the SSSP problem.

Chapter 7 Conclusions and Future Work

The work in this thesis initially investigated the optimization of Staffing and Scheduling a Software Project (SSSP) problem. From the literature, we found that benchmarking and evaluating the approaches proposed to solve this problem has only been done in the context of a comprehensive survey and a systematic literature review. Therefore, a complexity classification with datasets corresponds to this classification were created to contribute to the SSSP literature. In addition, a process combined with a set of quality and performance measures were proposed. As these approaches are proposed to solve an industrial problems, nine well-known approaches were under investigation of their quality and suitability to software industry using the benchmarking process and the datasets. The insight gained from the findings of investigating these approaches has contributed in formalizing four team allocation methods into optimization problems. In addition, an empirical evaluation of Project Managers (PMs) performance from software industry was performed. Part of this evaluation was to assess the suitability of the SSSP approaches by validating the datasets used to benchmark and evaluate them.

The answers for the first and second questions outlined in Section 1.4 of the first aim of this research can be concluded as follow. With no prior knowledge about the SSSP approaches, is there an automated approach that reliably solves the SSSP problem. Many optimized approaches have been presented throughout the previous three decades as in [15, 22], and it is important to capture their potential capability and capacity for different management complexity problems. Work has been carried out for this thesis in exploring the capacity and capability for nine SSSP approaches. The findings according to the measurements used for the approaches' outcomes using MAAPE for accuracy, and CT score for performance show that for project time problem, some of the SSSP approaches vary in their outcomes of Estimated Project Time (EPT) and Computation Time (CT), and the SSSP approaches in [22] and [94] can outperform the others as they are capable of providing solutions close to the optimal one with reasonable amount of computation time. While the SSSP approaches are differing from each other, it was important to observe which of the

allocation methods adopted by the approaches is capable of providing better solutions. In this matter, formalization of four team allocation methods into optimization problems was proposed, and advanced experiments were performed, using uniform stochastic operations and optimization settings of GA, to capture which of the methods are best at handling complexity level of effort, dependency, skills, and productivity. The finding from these experiments is that Static Teams with Queueing Simulator for allocation (STQS) for scenario one and two, and Dynamic Team with Binary Participation (DTBP) methods, for scenario three, four, and five were good at enabling the approaches to heuristically search for near optimal solutions.

The second aim of this research was to answer whether these approaches outperform the expert intuition in solving SSSP problem. Accordingly, an industrial setting study was performed. PMs were the subjects in this study for experiments and interviews. Four of the PMs work for a large financial organization, one with a large international software development organization, and the last one with Start-up Company. Our study subjects have between 6 to 30 years of experience. This study encompassed mixed-methods to capture different quantitative and qualitative data important in providing comprehensive knowledge about the study subjects and industry practice. The experimental part was performed to capture how PMs from the industry would perform for each scenario defined in the benchmark. As these scenarios are based on the dataset created for this thesis, the optimal project time was defined for each. There was two subjects that their solutions were similar to the best SSSP approaches. The key differences identified between those subjects and the others might have contributed to their high quality solutions. These keys are the allocation method they have used while solving the scenarios, as by dynamic teams, and distinguished years of experience they have, for 26 and 30 years. On the other hand, the accuracy of the solution provided by the SSSP approaches has a negative relationship with the level of scenario's complexity. For instance, as the level of scenario's increases from one to four, the results were less and less accurate as from 99.9% to 79.3%. In addition, with variability of resources' productivity implemented in scenario five, none of the approaches were able to handle this level.

The third aim of this research, identified in Section 1.4, was to find whether the SSSP approaches reflect the real PMs' needs. For this aim, interviews with PMs were conducted in the second phase carried out for the industrial settings evaluation study. This study was also designed to explore based on the subjects' experience, what aspects and attributes in software projects are important to be considered by a PM for SSSP problem. Seven categories were the focal points to discuss with the PMs in the interview. These categories are the organization level they represent, their experience, project attributes, allocation methods, teams, scheduling, and management objectives they believed are important to resource allocation optimization. Based on the results from the

interviews and the experiments carried out for this thesis, almost all the project attributes are found important.

Three main attributes discussed by the PMs in the interviews and found important for a PM to consider for software project optimization are the precedence relationship between project tasks, resource's skillset, and resource's productivity. These attributes are represented in the study partially by the challenging scenario four and fully represented by scenario five. It is noteworthy that scenario five was not included within the industrial settings study for two reasons. The main reason is that although the experiments were intended to challenge the PMs' capabilities, these experiments should also respect their time constraints too. The second reason is that the intended experiments and interviews are carefully planned to capture some targeted issues that can provide glimpse on PMs' practices, so it is hard to bring all the scenarios, especially the fifth one, to be solved by the subjects while a similar can be found in the fourth.

The experiments carried out on the four team allocation methods complies with the findings from the second phase of the industrial settings evaluation, which implies that the PM should look at the different resource assignment and team allocation methods that (s)he can use. In this regard, there was a consensus between the subjects on the dynamic shifting of resources between teams and tasks especially when a skill is required. Dynamic assignment method has been evaluated with different simulation of methods, and it was found from the demonstration of the simulations presented in Chapter 5 that this method can outperform the others and solve the time problem more efficiently to more advance scenarios.

Project cost, on the other hand, is clearly an important part that should be included within the optimization problem of resource allocation. However, as searching for the most minimized cost of resource allocation alternative requires the identification of resources' salary, the resources who are possessing the same skillset and doing the same job should have the same salary. Therefore, for an optimization problem that consider skills as one of the inputs, all the alternatives could have the same influence on project cost. The datasets used in this thesis moreover, have no information that can support this part due to the sensitivity of this information to the data contributors, which can show their key success in resourcing and payment structure. In addition, many SSSP approaches optimize either for time span or combine multi-objectives problem that includes project time within. Consequently, uniformalising the comparison problem for a shared objective –as this was the first intention of this thesis- of time span optimization problem is the only solution. Therefore, the main focus of the experiments carried out for this thesis was on project time span minimization using the described project in the datasets. Conforming to this conclusion the study subjects have stated in reaction to when the solutions of the scenarios was

revealed to them in the second phase that the important objective is time to how it can be managed after the agreement on quality and cost outcomes is established. To this end, it important to mention that four out of six subjects in the exit interviews stated that time is the ultimate objective in software projects.

7.1 Overall Findings and Lessons Learned

The overall findings throughout the thesis work are listed in the following bullet points.

- Different methods can be used to allocate human resource in software projects adopted by the PMs, and yet no specific method can overcome project time optimization problem.
- Teams are the solid base for software development, however, the assignment of team members can be represented by a rigid or dynamic formation and mainly less experience PMs tend to use the rigid one to avoid any conflicts.
- The more experienced the PM is, the more (s)he tends to provide better and faster solution to the problem.
- Complexity levels corresponding to PMs' projects problem are found by challenging scenario 2, and 4.
- Corresponding to the discussion made in [130], the work carried out for this thesis found that large organizations prefer a hybrid approach supported with a traditional development methodology, such as waterfall.
- Project time span is important to be managed as software cost and quality are issues that are defined in early project stages and the PM is left to manage the time according to the stakeholder's agreement.
- None of the of SSSP approaches has presented their experiments computation time for a matter of fact and many have failed to report the crucial system capability they used such as CPU, and memory.

In addition, there are some lesson learned as a result from the experiments and industry settings evaluation study are:

- It is hard to establish a communication with software industry for data to share.
- There is no available dataset that can be used for this filed of research and many are using hypothetical data.
- It is hard to bring more than one PM to an interview or experiment due to their work and time availability constraints, and that's can make it hard for a researcher to standardise the experiment interpretations and meanings exactly the same to all subjects. That was

the reason for having exit interviews after the experiments, so to make sure that they have understood and solve the scenarios to the best they can do and as we expect.

- Solutions to labour cost are mainly developed by the optimization approaches as in [93, 136] over time window and activity timing concepts. And this what has stopped us from including more approaches for SSSP.

7.2 Limitations and Future Work

The overall work carried out for this thesis has shed the light on some areas where improvements can be made. One of the improvements that needs to be done on the benchmarking side of SSSP approaches concerns the limitation of the datasets used in this thesis. These datasets present single problem of a small project with limited variables and information to a single organization, which can make it harder to conclude their applicability for different organizations and project scenarios. Moreover, there is a lack of representative datasets that includes the important parts and project attributes for SSSP optimization and none is freely available. Accordingly, gathering and collecting data from larger software projects is an important part in this research field to be made.

Different open source optimization toolboxes have been used by SSSP optimization papers, and yet no quality and performance information can be found on their outcomes compared against each other. Therefore, work has to be done on investigating and benchmarking these toolboxes especially for accuracy and computation time performance compared to the one used in the work for this thesis.

Moreover, the work carried out for this thesis has included nine approaches for specific problem comparison, however, work should be made to include up-to-date SSSP approaches in the benchmarking study. This work should investigate the novelty of the proposed approach and whether better outcomes are anticipated by the optimized allocation method proposed in that approach. In particular, benchmarking and comparing between the approaches adopting event-based, time-line, and time window scheduler are also important to be established to demonstrate their effectiveness and performance against each other. That includes developing a benchmark dataset for this particular approaches' type too. These approaches can help to tackle resource availability problem by providing, on the project time-line base, when the possible shortage of resources can happen, and their availability that can be supported by limiting their participation rate percentage.

On the other hand, another improvement that need to be done in the empirical side of software project time optimization is to expand the study to include different software organizations for more experimentation and interviews with different PM levels. Unlike the study presented in Chapter 6, PMs should be asked to perform their allocation to the challenging scenarios with a proper control so all the subjects have the same level of clarification and explanations, and any misunderstanding or misleading terminology can be avoided. This can be tackled by allocating different session dates and times supported by team of researchers to provide uniform description across these sessions. This should help to whether confirm the findings from the industrial settings study performed for this thesis or, to explore more different team formation and allocation practices of PMs, and the different problem representations they consider. More reliable outcomes with uniform problem formalization to common software projects are anticipated by this study to help create an optimization tool that can support the PM on his/her management task. This optimization tool can combine the additional aspects identified by the industrial settings study subjects presented in Chapter 6. This study concludes four pivotal aspects important for SSSP optimization approach to consider are cost and time objectives, availability of resources to each task, multi-project environment, and the formation of cross-functional teams. Our intention accordingly is to combine these aspects and parameters, and encompasses them within an optimization approach.

Learning effects moreover, can be another parameter for an optimization approach to combine within its process for estimating project time span. The relationship between different technical skills and software development competencies can provide identification of when and to which task the resource's productivity can be improved. This consideration of skills and competencies has been addressed in different incarnation as in [57, 62], however, it has never been addressed within an optimized approach. Therefore, work has to be accomplished towards understanding the relation between software development skills and competencies, and the combination of these skills and competencies with the learning effects into an optimized approach.

Bibliography

1. Suarez, L.F. *Resource allocation: a comparative study*. 1987. Project Management Institute.
2. Tsai, H.-T., H. Moskowitz, and L.-H. Lee, *Human resource selection for software development projects using Taguchi's parameter design.* European Journal of Operational Research, 2003. **151**(1): p. 167-180.
3. PMI, *A guide to the project management body of knowledge*, in *5th edn*. 2013, Project Management Institute, Newtown Square, PA.
4. Ruhe, G. and C. Wohlin, *Software Project Management: Setting the Context*, in *Software Project Management in a Changing World*. 2014, Springer. p. 1-24.
5. Ferrucci, F., M. Harman, and F. Sarro, *Search-Based Software Project Management*, in *Software Project Management in a Changing World*, G. Ruhe and C. Wohlin, Editors. 2014, Springer Berlin Heidelberg. p. 373-399.
6. Brooks Jr, F.P., *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, 2/E*. 1995: Pearson Education India.
7. Otero, L.D., et al., *A systematic approach for resource allocation in software projects.* Computers & Industrial Engineering, 2009. **56**(4): p. 1333-1339.
8. Chow, T. and D.-B. Cao, *A survey study of critical success factors in agile software projects.* Journal of Systems and Software, 2008. **81**(6): p. 961-971.
9. Lamersdorf, A., J. Münch, and D. Rombach. *A survey on the state of the practice in distributed software development: Criteria for task allocation*. in *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*. 2009. IEEE.
10. Harman, M. and B.F. Jones, *Search-based software engineering.* Information and Software Technology, 2001. **43**(14): p. 833-839.
11. Harman, M., S.A. Mansouri, and Y. Zhang, *Search-based software engineering: Trends, techniques and applications.* ACM Computing Surveys (CSUR), 2012. **45**(1): p. 11.
12. Chang, C., et al. *Spmnet: a formal methodology for software management*. in *Computer Software and Applications Conference, 1994. COMPSAC 94. Proceedings., Eighteenth Annual International*. 1994. IEEE.
13. Bagnall, A.J., V.J. Rayward-Smith, and I.M. Whittley, *The next release problem.* Information and software technology, 2001. **43**(14): p. 883-890.
14. Alba, E. and J.F. Chicano, *Software project management with GAs.* Information Sciences, 2007. **177**(11): p. 2380-2401.

15. Minku, L.L., D. Sudholt, and X. Yao, *Improved evolutionary algorithm design for the project scheduling problem based on runtime analysis.* IEEE Transactions on Software Engineering, 2014. **40**(1): p. 83-102.
16. Barreto, A., M.d.O. Barros, and C.M. Werner, *Staffing a software project: A constraint satisfaction and optimization-based approach.* Computers & Operations Research, 2008. **35**(10): p. 3073-3089.
17. Harman, M., et al., *Search based software engineering: Techniques, taxonomy, tutorial*, in *Empirical software engineering and verification*. 2012, Springer. p. 1-59.
18. Chang, C.K., M.J. Christensen, and T. Zhang, *Genetic algorithms for project management.* Annals of Software Engineering, 2001. **11**(1): p. 107-139.
19. Ibaraki, T., T. Kameda, and N. Katoh, *Cautious transaction schedulers for database concurrency control.* IEEE transactions on software engineering, 1988. **14**(7): p. 997-1009.
20. Antoniol, G., M. Di Penta, and M. Harman. *A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty*. in *Software Metrics, 2004. Proceedings. 10th International Symposium on*. 2004. IEEE.
21. Antoniol, G., M. Di Penta, and M. Harman. *Search-based techniques applied to optimization of project planning for a massive maintenance project*. in *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*. 2005. IEEE.
22. Di Penta, M., M. Harman, and G. Antoniol, *The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study.* Software: Practice and Experience, 2011. **41**(5): p. 495-519.
23. Park, J., et al., *Human Resource Allocation in Software Project with Practical Considerations.* International Journal of Software Engineering and Knowledge Engineering, 2015. **25**(01): p. 5-26.
24. Peixoto, D.C., G.R. Mateus, and R.F. Resende. *The Issues of Solving Staffing and Scheduling Problems in Software Development Projects*. in *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*. 2014. IEEE.
25. Patterson, J.H., *A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem.* Management science, 1984. **30**(7): p. 854-867.
26. Bibi, N., Z. Anwar, and A. Ahsan, *Comparison of Search-Based Software Engineering Algorithms for Resource Allocation Optimization.* Journal of Intelligent Systems, 2016. **25**(4): p. 629-642.
27. Sajad, M., et al., *Software Project Management: Tools assessment, Comparison and suggestions for future development.* International Journal of Computer Science and Network Security (IJCSNS), 2016. **16**(1): p. 31.

28. Kang, D., J. Jung, and D.H. Bae, *Constraint-based human resource allocation in software projects.* Software: Practice and Experience, 2011. **41**(5): p. 551-577.
29. Khatib, S.M.A. and J. Noppen, *Benchmarking and Comparison of Software Project Human Resource Allocation Optimization Approaches.* SIGSOFT Softw. Eng. Notes, 2017. **41**(6): p. 1-6.
30. Albrecht, A.J. and J.E. Gaffney, *Software function, source lines of code, and development effort prediction: a software science validation.* Software Engineering, IEEE Transactions on, 1983(6): p. 639-648.
31. Shepperd, M. and C. Schofield, *Estimating software project effort using analogies.* Software Engineering, IEEE Transactions on, 1997. **23**(11): p. 736-743.
32. Albrecht, A.J. *Measuring application development productivity*. in *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*. 1979.
33. IFPUG, F., *International Function Point Users Group (IFPUG) Function Point Counting Practices Manual*. 2000, Release.
34. Boehm, B.W., *Software engineering economics.* 1981.
35. Boehm, B. and W. Royce, *Ada COCOMO and the Ada process model*. 1989, DTIC Document.
36. Boehm, B., et al., *Cost models for future software life cycle processes: COCOMO 2.0.* Annals of software engineering, 1995. **1**(1): p. 57-94.
37. Paulk, M., *Capability maturity model for software*. 1993: Wiley Online Library.
38. Banker, R.D., H. Chang, and C.F. Kemerer, *Evidence on economies of scale in software development.* Information and Software Technology, 1994. **36**(5): p. 275-282.
39. Dillibabu, R. and K. Krishnaiah, *Cost estimation of a software product using COCOMO II. 2000 model–a case study.* International Journal of Project Management, 2005. **23**(4): p. 297-307.
40. JØRGENSEN, M. and K. MOLØKKEN-ØSTVOLD. *A review of surveys on software effort estimation*. in *International Symposium on Empirical Software Engineering (ISESE'03), Rome. Proceedings... IEEE Computer Society*. 2003.
41. Larrabee, R., *Effective Work Breakdown Structures [Book Review].* Software, IEEE, 2003. **20**(2): p. 84-85.
42. Haugan, G.T., *Effective work breakdown structures*. 2002: Management Concepts Inc.
43. Tausworthe, R.C., *The work breakdown structure in software project management.* Journal of Systems and Software, 1980. **1**: p. 181-186.
44. Parkinson, C.N. and R.C. Osborn, *Parkinson's law, and other studies in administration*. Vol. 24. 1957: Houghton Mifflin Boston.
45. West, D.B., *Introduction to graph theory*. Vol. 2. 2001: Prentice hall Upper Saddle River.
46. Malcolm, D.G., et al., *Application of a technique for research and development program evaluation.* Operations research, 1959. **7**(5): p. 646-669.

47. Kelley Jr, J.E. and M.R. Walker. *Critical-path planning and scheduling*. in *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*. 1959. ACM.
48. Gantt, H.L., *Work, wages, and profits: their influence on the cost of living*. 1910: Engineering magazine.
49. Wilson, J.M., *Gantt charts: A centenary appreciation.* European Journal of Operational Research, 2003. **149**(2): p. 430-437.
50. Clark, W., *The Gantt Chart.* The Ronald Press Co, New York 1925.
51. Badiru, A.B., *Activity-resource assignments using critical resource diagramming.* Project Management Journal, 1993. **24**: p. 15-15.
52. Milatovic, M. and A.B. Badiru, *Applied mathematics modeling of intelligent mapping and scheduling of interdependent and multi-functional project resources.* Applied mathematics and computation, 2004. **149**(3): p. 703-721.
53. Boucher, X., E. Bonjour, and B. Grabot, *Formalisation and use of competencies for industrial performance optimisation: A survey.* Computers in industry, 2007. **58**(2): p. 98-117.
54. INFOCOMP, *Information Technology Competency Model.* Employment and Training Administration, United States Department of Labour, http://www.careeronestop.org/competencymodel/pyramid_download.aspx?IT=Y., 2012.
55. SWECOM, *Software Engineering Competency Model* IEEE, http://www.computer.org/portal/web/pab/SWECOM, 2014.
56. Ardis, M.A. and P.B. Henderson, *Software engineering education (SEEd).* ACM SIGSOFT Software Engineering Notes, 2014. **39**(4): p. 11-12.
57. Acuna, S.T., N. Juristo, and A.M. Moreno, *Emphasizing human capabilities in software development.* Software, IEEE, 2006. **23**(2): p. 94-101.
58. Moses, J.L., *Applying the assessment center method*. Vol. 71. 1977: Pergamon.
59. Russell, M.T., et al., *16PF Fifth Edition administrator's manual*. 1994: Institute for Personality and Ability Testing, Incorporated.
60. Moses, J.L. and W.C. Byham, *Applying the Assessment Center Method: Pergamon General Psychology Series*. 2013: Elsevier.
61. Costa, P.T. and R.R. MacCrae, *Revised NEO Personality Inventory (NEO PI-R) and NEO Five-Factor Inventory (NEO FFI): Professional Manual*. 1992: Psychological Assessment Resources.
62. André, M., M.G. Baldoquín, and S.T. Acuña, *Formal model for assigning human resources to teams in software projects.* Information and Software Technology, 2011. **53**(3): p. 259-275.

63. Myers, I.B., et al., *MBTI manual: A guide to the development and use of the Myers-Briggs Type Indicator*. Vol. 3. 1998: Consulting Psychologists Press Palo Alto, CA.
64. Henry, S.M. and K.T. Stevens, *Using Belbin's leadership role to improve team effectiveness: An empirical investigation.* Journal of Systems and Software, 1999. **44**(3): p. 241-250.
65. Girba, T., et al. *How developers drive software evolution*. in *Principles of Software Evolution, Eighth International Workshop on*. 2005. IEEE.
66. Moura, M.H.D.d., H.A.D.d. Nascimento, and T.C. Rosa. *Extracting new metrics from Version Control System for the comparison of software developers*. in *Software Engineering (SBES), 2014 Brazilian Symposium on*. 2014. IEEE.
67. Royce, W., *Software project management*. 1998: Pearson Education India.
68. Jensen, R.W., L. Putnam, and W. Roetzheim, *Software estimating models: three viewpoints.* CrossTalk, 2006. **19**(2): p. 23-29.
69. Kitchenham, B.A., E. Mendes, and G.H. Travassos, *Cross versus within-company cost estimation studies: A systematic review.* Software Engineering, IEEE Transactions on, 2007. **33**(5): p. 316-329.
70. Jørgensen, M. and B. Boehm, *Software Development Effort Estimation.* 2009.
71. Jorgensen, M., *What We Do and Don't Know about Software Development Effort Estimation.* Software, IEEE, 2014. **31**(2): p. 37-40.
72. Lawler, E.L. and D.E. Wood, *Branch-and-bound methods: A survey.* Operations research, 1966. **14**(4): p. 699-719.
73. Rothlauf, F., *Design of modern heuristics: principles and application*. 2011: Springer Science & Business Media.
74. Gendreau, M. and J.-Y. Potvin, *Metaheuristics in combinatorial optimization.* Annals of Operations Research, 2005. **140**(1): p. 189-213.
75. Clausen, J., *Branch and bound algorithms-principles and examples.* Department of Computer Science, University of Copenhagen, 1999: p. 1-30.
76. Pardalos, P.M. and M.G. Resende, *Handbook of applied optimization*. 2001: Oxford University Press.
77. EPK, I.T., *Foundations of constraint satisfaction*. 1993, Academic Press Ltd., London.
78. Grötschel, M. and O. Holland, *Solution of large-scale symmetric travelling salesman problems.* Mathematical Programming, 1991. **51**(1-3): p. 141-202.
79. Gutin, G., A. Yeo, and A. Zverovich, *Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP.* Discrete Applied Mathematics, 2002. **117**(1): p. 81-86.
80. Cormen, T.H., *Introduction to algorithms*. 2009: MIT press.
81. Davis, L. *Bit-climbing, representational bias, and test suit design*. in *Proc. Intl. Conf. Genetic Algorithm, 1991*. 1991.

82. Russell, S. and P. Norvig, *Artificial intelligence: a modern approach.* 1995.
83. Holland, J.H., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. 1992: MIT press.
84. Falkenauer, E., *Genetic algorithms and grouping problems*. 1998: John Wiley & Sons, Inc.
85. Goldberg, D.E. and K. Deb, *A comparative analysis of selection schemes used in genetic algorithms.* Foundations of genetic algorithms, 1991. **1**: p. 69-93.
86. Mitchell, M., *An introduction to genetic algorithms*. 1998: MIT press.
87. Konak, A., D.W. Coit, and A.E. Smith, *Multi-objective optimization using genetic algorithms: A tutorial.* Reliability Engineering & System Safety, 2006. **91**(9): p. 992-1007.
88. Deb, K., et al., *A fast and elitist multiobjective genetic algorithm: NSGA-II.* Evolutionary Computation, IEEE Transactions on, 2002. **6**(2): p. 182-197.
89. Amouzgar, K., *Multi-objective optimization using Genetic Algorithms.* 2012.
90. Eberhart, R.C. and J. Kennedy. *A new optimizer using particle swarm theory*. in *Proceedings of the sixth international symposium on micro machine and human science*. 1995. New York, NY.
91. Shao, B.B., P.-Y. Yin, and A.N. Chen, *Organizing knowledge workforce for specified iterative software development tasks.* Decision Support Systems, 2014. **59**: p. 15-27.
92. Kuo, R.J., C.M. Chao, and Y. Chiu, *Application of particle swarm optimization to association rule mining.* Applied Soft Computing, 2011. **11**(1): p. 326-336.
93. Chang, C.K., et al., *Time-line based model for software project scheduling with genetic algorithms.* Information and Software Technology, 2008. **50**(11): p. 1142-1154.
94. Ren, J., M. Harman, and M. Di Penta. *Cooperative co-evolutionary optimization of software project staff assignments and job scheduling*. in *International Symposium on Search Based Software Engineering*. 2011. Springer.
95. Harman, M., S.A. Mansouri, and Y. Zhang, *Search based software engineering: A comprehensive analysis and review of trends techniques and applications.* Department of Computer Science, King's College London, Tech. Rep. TR-09-03, 2009.
96. Ngo-The, A. and G. Ruhe, *Optimized resource allocation for software release planning.* Software Engineering, IEEE Transactions on, 2009. **35**(1): p. 109-123.
97. Whitley, D., *A genetic algorithm tutorial.* Statistics and computing, 1994. **4**(2): p. 65-85.
98. Boehm, B.W., R. Madachy, and B. Steece, *Software Cost Estimation with Cocomo II with Cdrom*. 2000: Prentice Hall PTR.
99. Costa, A.P.C.S., *Decision model for allocating human resources in information system projects.* International Journal of Project Management, 2013. **31**(1): p. 100-108.
100. Martin, R.C., *Agile software development: principles, patterns, and practices*. 2003: Prentice Hall PTR.

101. Erich, F., C. Amrit, and M. Daneva, *A mapping study on cooperation between information system development and operations*, in *Product-Focused Software Process Improvement*. 2014, Springer. p. 277-280.
102. Michalewicz, Z., *Evolutionary Programming and Genetic Programming*, in *Genetic Algorithms+ Data Structures= Evolution Programs*. 1996, Springer. p. 283-287.
103. Haupt, R.L., S.E. Haupt, and S.E. Haupt, *Practical genetic algorithms*. Vol. 2. 1998: Wiley New York.
104. Sim, S.E., S. Easterbrook, and R.C. Holt. *Using benchmarking to advance research: A challenge to software engineering*. in *Software Engineering, 2003. Proceedings. 25th International Conference on*. 2003. IEEE.
105. Kwok, Y.-K. and I. Ahmad, *Benchmarking and comparison of the task graph scheduling algorithms.* Journal of Parallel and Distributed Computing, 1999. **59**(3): p. 381-422.
106. Lessmann, S., et al., *Benchmarking classification models for software defect prediction: A proposed framework and novel findings.* IEEE Transactions on Software Engineering, 2008. **34**(4): p. 485-496.
107. Arcuri, A. and L. Briand. *A practical guide for using statistical tests to assess randomized algorithms in software engineering*. in *Software Engineering (ICSE), 2011 33rd International Conference on*. 2011. IEEE.
108. Demšar, J., *Statistical comparisons of classifiers over multiple data sets.* Journal of Machine learning research, 2006. **7**(Jan): p. 1-30.
109. Wallis, S., *Binomial confidence intervals and contingency tests: mathematical fundamentals and the evaluation of alternative methods.* Journal of Quantitative Linguistics, 2013. **20**(3): p. 178-208.
110. Kitchenham, B.A., et al., *Preliminary guidelines for empirical research in software engineering.* IEEE Transactions on software engineering, 2002. **28**(8): p. 721-734.
111. Kitchenham, B., et al., *Robust statistical methods for empirical software engineering.* Empirical Software Engineering, 2017. **22**(2): p. 579-630.
112. Kim, S. and H. Kim, *A new metric of absolute percentage error for intermittent demand forecasts.* International Journal of Forecasting, 2016. **32**(3): p. 669-679.
113. T. Menzies, B.C., E. Kocaguneli, J. Krall, F. Peters, and B. Turhan,, *The PROMISE Repository of empirical software engineering data* 2012
114. Wright, H.K., M. Kim, and D.E. Perry. *Validity concerns in software engineering research*. in *Proceedings of the FSE/SDP workshop on Future of software engineering research*. 2010. ACM.
115. Di Penta, M. and D.A. Tamburri. *Combining quantitative and qualitative studies in empirical software engineering research*. in *Proceedings of the 39th International Conference on Software Engineering Companion*. 2017. IEEE Press.

116. Makridakis, S., *Accuracy measures: theoretical and practical concerns.* International Journal of Forecasting, 1993. **9**(4): p. 527-529.
117. Myers, G.J., C. Sandler, and T. Badgett, *The art of software testing*. 2011: John Wiley & Sons.
118. Shen, X., et al., *Dynamic software project scheduling through a proactive-rescheduling method.* IEEE Transactions on Software Engineering, 2016. **42**(7): p. 658-686.
119. Miranda, E. and P. Bourque, *Agile monitoring using the line of balance.* Journal of Systems and Software, 2010. **83**(7): p. 1205-1215.
120. Bendoly, E., J.E. Perry-Smith, and D.G. Bachrach, *The perception of difficulty in project-work planning and its impact on resource sharing.* Journal of Operations Management, 2010. **28**(5): p. 385-397.
121. Kang, K. and J. Hahn, *Learning and forgetting curves in software development: Does type of knowledge matter?* ICIS 2009 Proceedings, 2009: p. 194.
122. Ferrucci, F., et al. *Not going to take this anymore: multi-objective overtime planning for software engineering projects*. in *Proceedings of the 2013 International Conference on Software Engineering*. 2013. IEEE Press.
123. Michalewicz, Z. and D.B. Fogel, *How to solve it: modern heuristics*. 2013: Springer Science & Business Media.
124. Soberón, P., *Problem-Solving methods in combinatorics*. 2013: Springer.
125. Antoniol, G., et al., *Assessing staffing needs for a software maintenance project through queuing simulation.* IEEE Transactions on Software Engineering, 2004. **30**(1): p. 43-58.
126. Moore, J.R. and N.R. Baker, *An analytical approach to scoring model design—Application to research and development project selection.* IEEE Transactions on Engineering Management, 1969(3): p. 90-98.
127. Hilton, M. and A. Begel, *A Study of the Organizational Dynamics of Software Teams.* 2018.
128. Stylianou, C. and A.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, 2016. **98**: p. 79-96.
129. Ebert, C. and S. Brinkkemper, *Software product management–An industry evaluation.* Journal of Systems and Software, 2014. **95**: p. 10-18.
130. Vijayasarathy, L.R. and C.W. Butler, *Choice of software development methodologies: Do organizational, project, and team characteristics matter?* IEEE Software, 2016. **33**(5): p. 86-94.
131. Butler, T., *Towards a hermeneutic method for interpretive research in information systems.* Journal of Information Technology, 1998. **13**(4): p. 285-300.

132. Maia, C.L.B., et al., *An evolutionary optimization approach to software test case allocation*, in *Computational Intelligence and Information Technology*. 2011, Springer. p. 637-641.
133. Britto, R., et al. *A hybrid approach to solve the agile team allocation problem*. in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. 2012. IEEE.
134. Myers, M.D., *Qualitative research in information systems.* Management Information Systems Quarterly, 1997. **21**(2): p. 241-242.
135. Dalcher, D., *Rethinking success in software projects: looking beyond the failure factors*, in *Software project management in a changing world*. 2014, Springer. p. 27-49.
136. Chen, W.-N. and J. Zhang, *Ant colony optimization for software project scheduling and staffing with an event-based scheduler.* Software Engineering, IEEE Transactions on, 2013. **39**(1): p. 1-17.
137. Kwok, Y.-K. and I. Ahmad, *Static scheduling algorithms for allocating directed task graphs to multiprocessors.* ACM Computing Surveys (CSUR), 1999. **31**(4): p. 406-471.

Appendix A

Related Document of Empirical Evaluation in Industrial Settings Study

The documents related to our empirical evaluation study are provided in this appendix. That includes the research information provided to participants in Section 1, research ethics approval in Section 2, participation consent in Section 3, exit-interview protocol 4, and interview questions in Section 5.

1. Research Information Sheet

Dear Sir, madam,

Who I am?

My name is Sultan Al Khatib a doctoral researcher from University of East Anglia under supervision of Dr Joost Noppen. My research is about the optimization of human resource allocation in software projects.

Why I need you?

The research is intended to evaluate a set of mathematical staffing and scheduling software project approaches to assess their relevance in an industrial settings. I would therefore like to compare these mathematical approaches against the resource allocation practice by experienced software project managers from the industry such as yourself. The goal is to assess whether practitioners can benefit from these mathematical approaches.

What benefit this research will gain from your participation and what benefit you will gain from this research?

Your participation is valuable with your resource allocation practice and experience. Based on your participation of your responses and feedbacks, practitioners get to see the benefit and downsides of the mathematical approaches. Practitioners can also benefit of improving their practice. The assessment of the accuracy and performance between your allocation and the mathematical approaches can lead you with benefiting a new managerial and mathematical approach to use, and this study will show which, how, and why a mathematical approach will be advised for use to software project managers.

There is no risk involved in participating as we will use synthetic data, and all we need is your time.

What you will be doing during this research?

This research is divided into two phases. The first phase consists of four resource allocation challenges. Each challenge poses an increasingly complex resource allocation. For each challenge provided, you are required to perform an allocation to the resources provided for the tasks described for the project, based on the information presented in each scenario. This part of the research is designed to take no longer than one hour to perform. We expect to provide us with your way of doing the allocation, and estimated project time according to your allocation. In addition, this part of the research will be sent to you so that during your availability of time you

can finish it up, and once all the scenarios are completed you can send it back to us. This is made to makes it easy for you in term of time, and relaxation.

Phase two consist of an exit interview which you can elaborate on your experience in the given challenges. Each participant will be interviewed to answer seven sections. Each section in this interview is focused on part of the managerial aspect surrounding the decision for doing the allocation. Aspect such as the organizational level, management experience, and team consideration can show how the provided scenarios on phase one are related to the need of software project managers. The output of this phase is to provide an insight about the applicability of the findings from the scenarios of phase one as well as to validate the mathematical approaches.

Approval to proceed with the evaluation, and consent to use your feedback?

Your participation is very valuable. The consent of this experiment can be found at the back of this document. However, if you feel uncomfortable to proceed you can ask at any time to stop. This will not affect your right to withdraw, cancel and/or delete any recording, written, and stored answers and feedbacks.

Contact detail:

Main researcher: Sultan Al Khatib

School of Computing Sciences
University of East Anglia
Norwich Research Park
Norwich NR4 7TJ
United Kingdom

Phone: +44 (0) 1603 593738
Mail: S.Al-Khatib@uea.ac.uk
Web: http://seg.cmp.uea.ac.uk/

Supervisor: Joost Noppen

School of Computing Sciences
University of East Anglia
Norwich Research Park
Norwich NR4 7TJ
United Kingdom

Phone: +44 (0) 1603 593738
Mail: j.noppen@uea.ac.uk
Web: http://seg.cmp.uea.ac.uk/

2. UEA Computing Science Research Ethics Committee Approval

Dear Sultan

The submission of your proposal has been considered by the UEA Computing Sciences Research Ethics Committee and I can confirm that your proposal has been approved.

Please could you ensure that any further amendments to either the protocol or documents submitted are notified to me, as Chair of CMP-REC, in advance and also that any adverse events which occur during your project are reported to the Committee.

The Committee would like to wish you good luck with your project.

Best wishes

Dan Smith
(Chair CMP-REC)

Approval reference: CMP/1617/R/13

Dr D.J. Smith
School of Computing Sciences
University of East Anglia
NORWICH NR4 7TJ, UK

email: Dan.Smith@uea.ac.uk
tel: +44 (0)1603 592608
room: SCI 2.04

University of East Anglia

UK Top 15 (Complete University Guide 2017)
Top 5 for student satisfaction (National Student Survey, 2005-2016)
World Top 1% (Times Higher Education World Rankings 2015-16)
World Top 100 for research excellence (Leiden Ranking 2016)

3. Participation Consent

Consent Form

Software Project Resource Allocation Optimization – Interview Consent Form

Researcher: Sultan Al Khatib

Participant: _____

Job Title: _____

Organisation: _____

Consent:

I hereby consent to participating in entrance, and exit interviews for the purposes made clear by the interviewer/researcher for the study of Software Project Resource Allocation Optimization. I am aware a recording and transcript will be made of interviews and that I may request a copy of these if desired. I also confirm I have received detailed information pertaining to this study and am aware that I can cease my participation at any time.

Signed: _____

Name: (Printed)_____

Date: _____/_____/_____

☐ I wish to remain anonymous in any output from the interviews.

☐ I give consent to use my words, to quote my statements, and to use recording for research*

and publication*.

☐ I would like a copy of the recordings* and transcripts* of my interviews.

* - *delete as appropriate*

4. Software Project Managers Interview Protocol

Introductory Protocol

To facilitate our note-taking, we would like to audio tape our conversations today. For your information, only researchers on the project will be privy to the tapes which will be eventually destroyed after they are transcribed. In addition, you must sign the consent form devised to meet our human subject requirements. Essentially, this document states that: (1) all information will be held confidential, (2) your participation is voluntary and you may stop at any time if you feel uncomfortable, and (3) we do not intend to inflict any harm. Thank you for your agreeing to participate.

We have planned this interview to last no longer than one hour. During this time, we have several questions that we would like to cover. If time begins to run short, it may be necessary to interrupt you in order to push ahead and complete this line of questioning.

Introduction

You have been selected to speak with us today because you have been identified as someone who has a great experience to share about software development, and software projects. Our research project as a whole focuses on comparing the automated mathematical models that optimize the software project resource allocation to most minimized project time, with particular interest in understanding how these academic approaches are engaged and close to the software industry needs. Our study does not aim to evaluate your techniques or experiences. Rather, we are trying to learn more about the adopted methods of resource allocation, and hopefully learn about best practices that help improve project managers in software industry.

Keys

(Open)	Question has this key means that the answer will be an open ended, with no restriction about number of words or statement the participant can give. The reason for this is that these question can be counted as a follow-up to the test provided for participants.
(Closed)	Question has this key means that the answer will be a close ended, with no restriction of Yes, or No answer. The reason for this is that these question can provided if the participants agree or disagree with what mentioned in the question.
(Probe)	Question has this key means according to the answer the interviewer will follow-up to gather further information.

(Intro)	Question has this key means the answer will only to ice breaking, and getting to know the participant which will aid in building a relationship between the interviewer and participant. This type of question will give an introductory information that only beneficial to understand the back ground of the participant. This type of question is likely to be at the starting and ending of the interview.

5. Software Project Managers Interview Questions

Institutions: _____

Interviewee (Title and Name): _____

Interviewer: _____

Date: _____

Interview Section Used:

_____ A: Organization Level

_____ B: Project Management Experience

_____ C: Project Attributes

_____ D: Software Project Resource Allocation

_____ E: Team Consideration

_____ F: Project Scheduling

_____ G: Resource Allocation Objective(s)

Other Topics Discussed: _____

_____

Documents Obtained: _____

_____

_____

Post Interview Comments or Leads:

_____

Exit Interview Questions

Organization Level:

1. **(Open, Intro)** How do you classify your firm/organization?

Question Note:
Capture the participant background and perspective of the software projects.

Project Management Experience:

1. **(Open, Intro)** How long have you been a project manager?

Question Note:
Capture the participant background of manging software projects.

2. **(Open, Intro)** What is the development methodology you use with your development teams?

Question Note:
Capture the participant background and perspective of managing software projects.

3. **(Open, Intro)** What project size do you classify yourself you have been working on since your start as PM?

Question Note:
Capture the participant background and perspective of managing software projects.

Project Attributes

1. **(Open, Probe)** Do you use any productivity measures of your resources, and if so what is it?

Question Note:
Capture whether the inputs adopted by the mathematical approaches are different from the subject practice and requirement.

2. **(Open, Probe)** Which one of the scenarios you think its attributes belong to the class of problem your organization have?

Question Note:
Capture the fitness of which scenario close to the participant practice and requirement.

3. **(Open, Probe)** Why these attributes you believe are important to use while performing the resource allocation?

Question Note:
Probing to gain clarification of the fitness of which scenario close to the participant practice and requirement.

Software Project Resource Allocation

1. **(Closed)** Do you use a clear method for allocating and assigning resources to your project tasks?

Question Note:
Capture the participant background and perspective of managing software projects.

2. **(Open, Probe)** What is the adopted method for allocating and assigning resources to project tasks, and is it for team or individual assignment to tasks?

Question Note:
Probing to gain clarification of participant practice, and reason for it.

3. **(Open, Probe)** What do you think of the dynamic assignment and allocation of resources either of distribution of resources into teams, or resource productivity change over time?

Question Note:
Probing to gain advanced clarification of the participant practice and requirement.

Project Team Consideration (answer this part if your answer is team to question 2 of project resources allocation)

1. **(open)** What criteria do you use to form a team, sharing similar competencies and skill, or creating a cross-functional one?

Question Note:
Probing to gain more understanding of the participant practice and requirement regarding team concept.

2. **(Closed)** Does the adopted team method considers technical or role attributes?

Question Note:
Probing to gain more understanding of the participant practice and requirement regarding team concept.

Project Scheduling

1. **(Open, Probe)** How do you consider dependencies between tasks while you allocating resources?

Question Note:
Capture the participant practice and requirement regarding scheduling problem.

2. **(Closed)** Do you think this problem should be seen from different angle, which should consider allocation of resource in a multi-project environment?

Question Note:
Capture the participant perspective, and practice regarding the scheduling problem.

3. **(Open, Probe)** In your perspective, what does that mean to consider dependencies between projects as well as the tasks of each project too?

Question Note:

Probing to gain more understanding of the participant practice and requirement regarding the scheduling problem.

Resource Allocation Objectives

1. **(open, Probe)** Do you consider minimizing the project time to be the ultimate objective of resource allocation, or in your projects you have to consider multiple objectives such as minimizing cost, team or resource utilization, maximizing team skills, etc.?

Question Note:
Capture views and adoption of resource allocation objectives.

2. **(open, Probe)** What are the important objective(s) of your resource allocation you think, if you have multiple objectives?

Question Note:
Probing to gain more understanding of the participant view, practice and/or requirement regarding the objectives of resource allocation.

3. **(Closed)** Do you think cost dimension has to be included within the objectives of solving resource allocation in software project while employees still have to be paid regardless their participation in projects?

Question Note:
Capture views and adoption of resource allocation objectives.

4. **(open, Probe)** If so, how do you think the cost should be calculated and do you think the cost of the software product is calculated based on the participation and salary of resources within the project – as adopted by the approaches so far?

Question Note:
Probing to gain more understanding of the participant practice and requirement regarding the method of costing software projects.

Appendix B

Benchmarking and Comparison of Software Project Human Resource Allocation Optimization Approaches Paper

This section presents our research paper, which presented in the doctoral symposium of Empirical Software Engineering and Measurement Conference (ESEIW). This paper is published in [29] by ACM SIGSOFT Software Engineering Notes.

Benchmarking and Comparison of Software Project Human Resource Allocation Optimization Approaches

Sultan M Al Khatib
School of Computing Sciences
University of East Anglia
Norwich Research Park
Norwich NR4 7TJ
United Kingdom
s.al-khatib@uea.ac.uk

Joost Noppen
School of Computing Sciences
University of East Anglia
Norwich Research Park
Norwich NR4 7TJ
United Kingdom
j.noppen@uea.ac.uk

ABSTRACT

For the Staffing and Scheduling a Software Project (SSSP), one has to find an allocation of resources to tasks while considering parameters such skills and availability to identify the optimal delivery of the project. Many approaches have been proposed that solve SSSP tasks by representing them as optimization problems and applying optimization techniques and heuristics. However, these approaches tend to vary in the parameters they consider, such as skill and availability, as well as the optimization techniques, which means their accuracy, performance, and applicability can vastly differ, making it difficult to select the most suitable approach for the problem at hand. The fundamental reason for this lack of comparative material lies in the absence of a systematic evaluation method that uses a validation dataset to benchmark SSSP approaches. We introduce an evaluation process for SSSP approaches together with benchmark data to address this problem. In addition, we present the initial evaluation of five SSSP approaches. The results shows that SSSP approaches solving identical challenges can differ in their computational time, preciseness of results and that our approach is capable of quantifying these differences. In addition, the results highlight that focused approaches generally outperform more sophisticated approaches for identical SSSP problems.

Keywords

Human Resource Allocation; Software Project Management; Optimization Techniques in Software Engineering; Comparative Study; Performance Evaluation

INTRODUCTION

Software development is a mixture of complex activities and the creation of any non-trivial software system generally requires multiple resources with a mix of skills, expertise, and knowledge. The assignment of those resources in a software development department to projects and tasks within those projects is one of the most critical tasks for a project manager, with limited resources, dependent tasks, and available skillsets needing to be considered to achieve an optimal project delivery time. This problem of staffing and scheduling a software project (SSSP) in order to minimize the project completion time has been attracting researchers since the end of last century [2, 5, 22, 24] and different optimization techniques have been used to address it in various incarnations [5, 14, 18]. These approaches typically consider specific attributes when optimizing the resource allocation such as task length, resource availability or skills, and the traversal of the optimization space is typically performed by using exact, heuristic, and meta-heuristic techniques in order to deal the NP-Complete nature of the allocation problem [5]. Project managers typically can select an automated SSSP approach to support their allocation process based on the project and resource properties they wish to consider. However, approaches can have different performance characteristics such as the accuracy of the allocation results or computational time required, characteristics that are critical for successful SSSP but very hard to determine without a systematic manner. Limited number of studies in this context [5, 24] were published that compare SSSP approaches but neither of these studies performs an empirical evaluation of SSSP approaches using a unified basis and data set.

This article proposes to address that gap by introducing a benchmark and using it to evaluate the performance of a set of SSSP approaches against well-defined performance measures. Specifically, we aim to provide a validation dataset that has both resources and detailed project information for a range of SSSP challenges. In addition, we aim to compare the SSSP approaches using a uniform and expandable set of performance measures that can compare SSSP approaches in various categories and supporting a range of optimization criteria.

In addition to the benchmark and initial results of the comparison analysis in this article, we also outline our research agenda. To further the accuracy and relevance of the performance evaluation we aim to perform a comparison of computational approaches and current industry standards. This will be complemented with the implementation and evaluation of additional SSSP approaches to form a complete and comprehensive overview of SSSP approaches as well as the means to perform systematic comparisons between them. Note that this should not be confused with the comparison of the heuristic algorithms. The comparison adopted in this paper considers the approaches that propose a model for allocating the developers in software projects with modification on the algorithms they use.

The remainder of this paper is organized into five sections. Section 2 describes the studies carried out in comparing SSSP approaches that are related to the work presented in this paper. Section 3 detailed the workflow of procedures, dataset, criteria proposed to evaluate and compare the SSSP approaches, future plan of carrying out the rest of study work, and the threats and weaknesses that could affect the validity of this study. In section 4, the approaches adopted in this study are described and the results of the experiments and comparison between the SSSP approaches are shown. Section 5 discusses the main findings and concludes the paper.

RELATED WORK

When considering previous work performed in the area of evaluating SSSP approaches, only two studies have been published that compare and evaluate the optimization approaches of SSSP. Both comparison studies were based on evaluating the approaches according to the description provided within the texts. These studies have compared the approaches by a comprehensive survey [3] or systematic literature review [4] by extracting the text that describing the problem and solution of the approaches. Thus, these studies are more formally systematic literature review with comprehensive survey of wide software project management approaches.

The first study by Pixoto et al [24] evaluates the solution provided by SSSP approaches regarding their applicability in real-world software development projects. Criteria used by Pixoto et al to evaluate the description of solutions are usefulness, work compatibility, and ease of use attributes. 52 approaches were considered by this study. The comparison shows that few approaches among them all are satisfying the criteria adopted and capable for the illustrated aspects by this study as the one in [28]. Skills and productivity of resources found are the least aspects considered by the approaches used by Pixoto et al [24]. In addition, time and cost of software projects are the goals adopted by overwhelming majority of SSSP approaches. It is also noticeable in this study that only 8% of the approaches compared found they have used experiments to validate their solution. The overall conclusion by this study is that more research is needed to bridge the gap between the current practices of software firms and the proposed solutions. As this study provides essential aspects and differences between the SSSP approaches, the adoption model of criteria and aspects used are based on theoretical models. Criteria and aspects however have to be validated by the industry before they can make their claims about the usefulness of the approaches used in their study.

The second study presented in Ferucci et al [5] provides a comprehensive survey of the approaches use optimization techniques to solve software project management problems. Their observations and findings highlight the categories of the optimization approaches, the important attributes that these approaches adopted, and the approaches that match their criteria and seen useful to be adopted. The approaches used by this study are categorized into minimizing project time, risk-based, overtime planning, and effort estimation. This study has also identified the future trends and promising areas of resource allocation optimization. The areas found require more attention by researchers as future trends are interactive optimization, dynamic adaptive optimization, multi-objective optimization, co-evolution, software project benchmarking, confident estimates, and decision support tools. While this study is a comprehensive survey, it can be seen as a general study that reports the different types of problems adopted by approaches deal with software project management with no consideration of further classification or either cross functionality between the approaches and how each has opened a new knowledge.

The results presented in these studies are a valuable insight into the relation between various SSSP approaches, however neither study performs a systematic comparison between the SSSP approaches considered based on their implementation and a reference dataset. This is due to the fact that a

benchmark dataset currently is not available in this research area. While two repositories exist for the use of software engineering research, which are ISBSG and Tera-PROMISE, none of these includes a valid dataset containing human resource models and detailed project information usable for SSSP based research [5]. Accordingly, there is an urgency in this particular area for a data that represent a real software project to benchmark the SSSP approaches [3]. As a result, comparing and benchmarking SSSP approaches based on their behaviour and performance has not been carried out even when it has been identified as highly important by the community [5].

A SYSTEMATIC APPROACH FOR COMPARING SSSP APPROACHES

Overview of the Proposed Approach

Our proposed approach for performing a systematic and reproducible performance comparison of SSSP approaches consists of a systematic sequence of steps to be followed combined with an evaluation dataset and a suite of evaluation criteria on which the SSSP approaches can be compared. The proposed workflow for evaluating a set of SSSP approaches consists of the following steps:

1. Select a set of candidate SSSP approaches that are capable of solving a resource allocation problem and belong to the same class – see section 3.2 -.
2. Select the suitable dataset from the benchmark dataset that belong to the same class of the approaches selected containing the desired resource and project properties (e.g. skills, task dependencies, etc.)
3. Run each approach for the configured dataset for a substantial number of times, (e.g 100 times).
4. Record for each run the result of estimated project time, and the computation time of that run (see below).
5. Compile the results and measure their performance using the benchmark metric suite (see below).
6. Rank the candidate SSSP approaches based on their score in the overall scoring model (see below).

These steps are depicted in Figure 1. As can be seen in the Figure, after identifying the approaches, the classes that they belong to, and selecting the suitable benchmark dataset, the datasets located on the left down of the figure is fed into each approach. As most approaches perform heuristic optimization using a probabilistic optimizer, step 3 suggests to perform multiple runs for each of those approaches so that their computation time and accuracy can be averaged, as well as their mean and standard deviation can be determined. The choice for these metrics is motivated by the fact that they are seen as the most useful way to represent effectiveness and performance among the approaches [105].

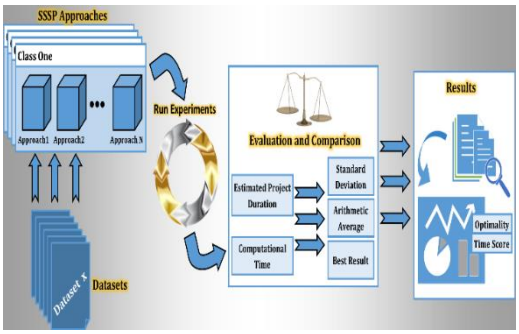

Figure 37: Proposed Approach

Benchmark Dataset

The first artefact we introduce to perform a systematic evaluation of SSSP approaches is a flexible and configurable benchmark dataset. The dataset is a small real world data from a Jordanian software company and holds information regarding both software project and human resources used to develop that software. This data includes information about eight components of the software projects, and twelve human resources were available to that project assigned to complete it. The project represented in the dataset has an estimated time using COCOMO. The time estimated with those resources available was 75.16 days, with an estimated Man-Day equals to 964. The dataset is composed of five sets the first four correspond to the classification made to the SSSP approaches. The first four sets describe resource allocation problems of increasing complexity and parameters. The final set describes a resource allocation problem of a larger size that is intended to analyse the scalability of the approaches in class 1. In addition, for each one of these classes the optimal solution (referred to as *min* value) as well as the worst-case solution values (referred to as *max* value). The dataset used in this article can be found on http://seg.cmp.uea.ac.uk/projects/resource-optimisation/files/dataset.zip.

When benchmarking SSSP approaches, it is critical to note that proposed approaches generally solve different variations of the resource allocation problem, taking into account different parameters, such as worker skills, or tasks dependencies. To evaluate the relative performance of SSSP approaches they need to be

applied to the same problem with the exact same inputs, which is why we propose to group SSSP approaches into classes according to the inputs and constraints required by each. The inputs required for resource allocation can be the estimated effort of project tasks, task dependencies, skills, and/or resource productivity. Each one of these inputs represented in the dataset by numbers except the skills. Skills required for developing each task or offered by a resource are representing languages and technologies, and represented in the dataset using the name of this language or technology such as java, or UML. Estimated effort of each task is represented by person-day. Each task in the dataset moreover has the value of dependency attribute represented as the task number that the task is depends on. Productivity of a resource is represented by the same metric used by [7]. A resource can be productive as a normal person, which is equal to 1, or twice the normal person represented by 2. According to these inputs the proposed classes are:

- **Class One**. This class contains the approaches that require inputs only of estimated effort of project tasks and the number and productivity of human resources.
- **Class Two**. This class contains the approaches that require inputs of estimated effort of project tasks, dependencies between these tasks, and number and productivity of human resources
- **Class Three**. This class contains the approaches that require inputs of estimated effort of project tasks, skills required for each tasks, and number, skills, and productivity of human resources
- **Class Four**. This class contains the approaches that require inputs of estimated effort of project tasks, dependencies between these tasks, skills required for each tasks, and the number, skills, and productivity of human resources.

Note that some SSSP approaches can possibly be part of multiple classes as they are able to determine the optimal allocation of resources for simple as well as complex SSSP problems. The performance for such approaches can be compared to other approaches in both classes with respect to solving identical problems. The benchmark data follows this classification as it defines optimization challenges within these five distinct classes to facilitate the uniform comparison of SSSP approaches

Comparison Metrics and Overall Scoring Model

The performance of a SSSP approach is usually measured in terms of optimality, i.e. how close the approach gets to the true optimal solution [137]. However, this metric only provides a partial view. For example, many probabilistic optimizers, such as genetic algorithms, vary in the quality of solution they provide due to a randomised starting point and the computation time expended to them. Accordingly, both of resulted values from the approach for the objective function - which in this study is the estimated project time- and the computational time expended to produce the results are the main metrics of this comparison. In addition to the performance measures of optimal solution and computation time, behaviour of the approaches have to be recorded too. While each approach uses a modified version of optimization technique, it is important to capture stability and preciseness of the approach over multiple runs. The importance of having a multiple runs is due to the probabilistic nature of meta-heuristic algorithm search. This can be depicted by the standard deviation of multiple runs of both estimated project time and computational time. To get a more complete insight into the performance of SSSP approaches we propose to use the following metrics:

1. **Estimated Project Time (EPT)**. The first proposed metric is the estimated project time, i.e. the identified optimal result by an approach for each run.
2. **Computational Time (CT)**. Computation time is the time consumed by the system to perform the approach from the point of feeding the data to the time of identifying the (heuristically) optimal result.
3. **Standard Deviation (STDEV)**. This metric is the standard deviation among the collected EPT values. This metric is a useful indicator of whether an approach is robust and precise. As the standard deviation will quantify outcomes produced are closely grouped or not.
4. **Arithmetic average (Mean)**. The mean of values resulting for an SSSP approach over multiple runs.
5. **Minimal EPT**. The least possible value for estimated project time among the collected values over multiple runs.

Note that metrics such as STDEVB and mean require the performance of the approach to be determined over multiple runs so that the average behaviour can be established and compared.

In addition to this suite of metrics, we propose the use of an overall scoring model for easy comparison of SSSP approaches, consisting of two formulas. The first formula captures the accuracy of a SSSP approach using the following equation:

Optimality of solution = [1-[(V-min)/(max-min)]] x 100

This formula depicts how close the value calculated by a SSSP approach (*V*) is to the known optimal solution (*min*).This value is normalised using the known worst-case solution (*max*). Both the *min* and *max* values are included in the dataset for a given SSSP problem. In addition, a model for scoring the computational time performance of an approach is depicted by the following equation.

CTime Score = [Vct / Max (Class)]

In this formula Vct is the computation expended by approach V to solve the SSSP problem under consideration of Max(Class) which is the maximum computation required for all known SSSP approaches capable of solving this problem.

Research Agenda for Comparison Benchmark of SSSP approaches

The work described in this paper is a first step towards a systematic mechanism for evaluating SSSP approaches with respect to their performance and accuracy. The research plan from this point focuses on extending the SSSP benchmark method and evaluating its usability and applicability in an industrial setting. To this purpose, the research plan is divided into four parts:

- The first part is the refinement of the benchmark dataset to include more projects and resource data as well as a refined configuration mechanism that allows for easy configuration.
- Second we aim to extend the set of implemented and evaluation SSSP approaches to provide a comprehensive set of data points that researchers can use to compare their own approaches to.
- Thirdly, we aim to examine a mechanism that allows us to easily bridge the gap between SSSP approaches so users of the benchmark can more easily evaluate a range of SSSP approaches against a set problem with specific parameters.
- Finally, upon establishing a reasonable and balanced SSSP benchmarking process we will evaluate its suitability and relevance by means of empirical evaluation with industrial partners. The results of the experienced project managers in allocating resources to projects will be compared to SSSP approaches and their benchmarking results for this purpose.

Threats to Validity and Challenges in comparing SSSP approaches

One of the main threats to validity in this study is that the data collected represents a single use of allocation attributes of one software firm, which can have an implication regarding the validity of the comparison with the different styles adopted in the industry regarding the allocation, constraints, and the development method within these firms. However as the dataset used to compare the approaches is a real-world data, it represent a small project which might not be the common scenario in software firms and the capabilities offered by various types of SSSP approaches are not covered such as dealing with a massive software project. Moreover, extending it to cover the capabilities of SSSP approaches while at the same time remaining representative can be very challenging. Thus, we aim to ensure the relevance of the data, and the approaches by expanding the experiments with our industrial partners. A further threat to the relevance of our evaluation results is the limited detail provided by publications describing SSSP approaches. In many cases, vital elements of the approach are not described sufficiently and no reference implementation of the approach is provided for evaluation. We have addressed this threat in our approach by excluding approaches with incomplete descriptions that prevented us to implement it. Where possible we have liaised with the authors of the approach to clarify ambiguities and complement the publication.

BENCHMARK APPLICATION TO EXISTING SET OF SSSP APPROACHES

Overview

To assess the accuracy and suitability for our proposed approach and benchmark we have performed a preliminary study of five SSSP approaches in two different classes. The approaches focus on optimizing the software project time using meta-heuristic techniques such as Genetic Algorithm (GA) and Simulated Annealing (SA) while taking into account various parameters such as task dependency to find the optimal or near optimal project time. The reason for selecting these approaches in this comparison is based on the studies presented in [3, 4]. These approaches are presented in Table1 according to the class they belong to. The approaches have been classified according to the SSSP classes introduced in Section 3.2. The optimization techniques used by the approaches are Genetic Algorithms (GA) by [14, 18, 20, 21], and a modified version of Simulated Annealing (SA) called Accelerated SA by [28]. Both techniques are belong to the same search algorithm class called meta-heuristic.

Table 47: Approaches Classification

Class	One	Two	Three	Four	Five
Approach					
[20]	**X**				
[21]	**X**				
[28]	**X**		**X**		**X**
[18]		**X**			
[14]		**X**		**X**	

Work has been accomplished to classify the approaches described earlier according to the classes they can use. This table shows the applicability of dataset classes too for each approach described earlier.

Results

The results were obtained using the Matlab R2013a supported by Matlab Global Optimization Toolbox using Intel Core 2 quad 2.66 Ghz CPU. Each approach was executed 100 times to allow determination of mean and deviation values. The comparisons performed were between Di Penta et al [20], Di Penta et al [21] and Kang et al for the Class 1 benchmark data, and between Chan et al and Alba et al for the Class 2 dataset.

Results of the Class One Dataset Evaluation

The first results we present are for the Class 1 approaches [20],[21], and Kang et al [28]. The dataset used is the Class One dataset, which only considers tasks, resources and availability, and has an optimal solution of 80.33 for its project schedule. Figure 2 shows how each iteration for each approach resulted an EPT in term of days where the lowest value amongst the approaches is the one obtained by DiPenta et al [20]. Moreover, we can see that the approach in both DiPenta [20] and [21] were quite close to the estimate of COCOMO presented in Section 3.3.

The results obtained for Kang et al approach on the other hand is overestimating project time when compared to any one of the DePinta el al approaches. This is due to the allocation method adopted by Kang et al approach as it assigns single resources to tasks with least estimated effort, where those that have the biggest effort required are each assigned to two resources which results in a less accurate approximation. The numeric results for accuracy are given in Table 2. It is interesting to observe that DiPenta et al [20] is the most accurate and it has managed to identify the actual optimal solution (80.33) for the dataset task. DiPenta et al [21] has come close to finding the optimal solution but Kang et al struggled to come close. A graphical representation of this data as well as the behaviour over multiple runs can be found in Figure 2.

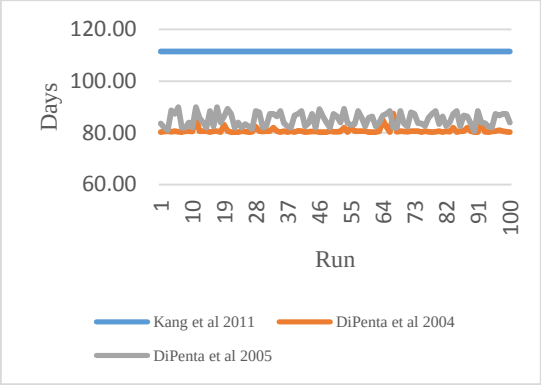

Figure 38: Accuracy performance over a 100 runs for Class One

When we examine the computation time results in Table 2. It can be seen that DiPenta et al [21] is the least time consuming among the approaches whereas Kang et al requires slightly more time. DiPenta et al [20] clearly requires the most time to identify an optimal solution.

Table 48: Performance results of Class One

Approach	Computation Time		Accuracy of Solution		
	Mean	STDEV	Mean	STDEV	Minimal EPT
[28]	127.90	2.82	111.5	0	111.5
[20]	285.91	2.57	80.83	1.139	80.33
[21]	109.65	0.19	85.13	2.61	80.6

An interesting observation as well is that while DiPenta et al [21] is not only faster, its standard deviation also is significantly lower than the two other approaches, which means the optimization behaves more uniformly in repeated experiments. This is a quality attribute that can become important when the problem size is scaled up, as a small variation in computation time can make solving a particular problem infeasible.

Results of the Class Two Dataset Evaluation

For the Class 2 approaches [14, 18] their performance was evaluated using the Class 2 dataset, where constraints are imposed on project schedule corresponding to dependencies between tasks. This

dataset has an optimal solution of 81.95 days for the project schedule. When examining the results in Table 3. It can be seen that the approach of Chang et al is capable of identifying the optimal solution where the approach by Alba et al is not, however the approach of Alba et al gives a more reliable and reproducible results for a single run, as illustrated by the standard deviation value. This becomes even more clear when examining Figure 3 where Chang et al clearly fluctuates per run where the results of Alba et al is more tightly grouped together.

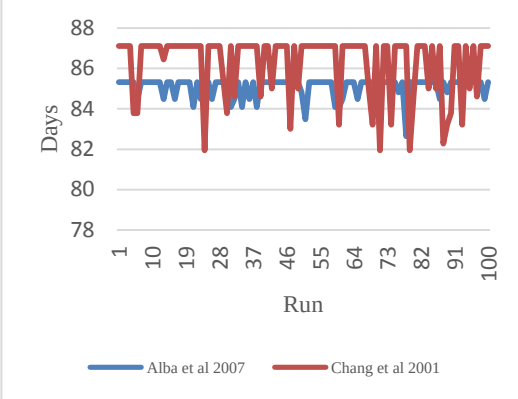

Figure 39: Accuracy performance over a 100 runs for Class Two

An interesting picture surfaces when we examine the computation time required by both approaches, as depicted in Table 3. It can be seen that while Chang et al fluctuates in the accuracy of the answer returned per run, on average it completes significantly faster than Alba et al. In this case, it is clear that while both approaches apply similar techniques Chang et al have sacrificed part of their accuracy for improved computation time performance.

Table 49: Performance results of Class Two

Approach	Computation Time		Accuracy of Solution		
	Mean	STDEV	Mean	STDEV	Minimal EPT
[18]	41.88	0.17	86.29	1.52	81.95
[14]	134.99	1.91	85.1	0.49	82.64

Ranking SSSP Approaches Comparison Using the Scoring Model

As the final step of our preliminary evaluation, we rank the evaluated SSSP approaches using our proposed scoring model. By combining the results of the approaches using the computation time and estimated project time and the formulas presented in Section3.3 we can compile the results in Table 4.

Table 50: Ranking results for the approaches

Class	Approach	Optimality of Result	CT Score
Class One	**[28]**	96.5%	0.45
	[20]	99.9%	1
	[21]	99.46%	0.3835
Class Two	**[18]**	99.37%	0.312
	[14]	99.54	1

This table gives an aggregated overview of the evaluation results using our dataset and metric suite. It can be seen for the Class 1 approaches that both approaches proposed by DiPenta et al are very close in accuracy but differ in computation time, with Kang et al representing a middle ground. For Class 2 a clearer winner can be identified with Chang et al offering similar accuracy to Alba et al but requiring far less time. We imagine that this aggregated scoring model will aid practitioners in comparing SSSP approaches and as such, it is one of the important deliverables of our research. Note however that in this scoring model at the moment the added value of standard deviation for both accuracy and computation is lost. In future work, we aim to include these explicitly in the scoring model to give a more complete picture.

CONCLUSIONS

In this article, we have identified that many different optimization approaches exist for staffing and scheduling a software projects (SSSP), but due to differences in the problem parameters they can consider as well as the optimization techniques they use their performance and applicability can be hard to assess and compare. To address this issue we have introduced a systematic comparison method for SSSP approaches together with a set of comparison metrics and an overall scoring model that can be used to rank their

performance. This comparison method is combined with a benchmark dataset and reference values that identifies and supports four different classes of SSSP approaches based on their capabilities and limitations. We have applied our method and benchmark data to a set of five SSSP approaches and from these early results the applicability and accuracy of our method became clear. Our method highlighted that focussed approaches that aim to solve a well-defined SSSP problem are more likely to identify an accurate solution within a reasonable amount of time rather than approaches that can potentially consider a wider range of parameters and inputs.

Our future work and the expected contribution of my dissertation lies first in the creation of a more comprehensive method and reference dataset for comparing SSSP approaches but also in evaluating this with industry experts who are expected to apply the method in practice. To achieve this we are planning further experiments and evaluation with the intention to expand the dataset and add support for the remaining SSSP classes. In addition, we aim to expand the range of SSSP problems per class in both complexity and size to aid in the evaluation of scalability. Finally, we aim to perform an empirical experiment where we ask industry experts to apply and evaluate various SSSP approaches and compare the results to the evaluation results of our method to establish the relevance and accuracy of the method in real-world application scenarios. Our eventual goal for this work is to serve as an accurate and flexible reference mechanism for both academics and practitioners for determining the performance and accuracy of SSSP approaches.

REFERENCES

1. Tsai, H.-T., H. Moskowitz, and L.-H. Lee, *Human resource selection for software development projects using Taguchi's parameter design.* European Journal of Operational Research, 2003. **151**(1): p. 167-180.
2. Di Penta, M., M. Harman, and G. Antoniol, *The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study.* Software: Practice and Experience, 2011. **41**(5): p. 495-519.
3. Ferrucci, F., M. Harman, and F. Sarro, *Search-Based Software Project Management*, in *Software Project Management in a Changing World*, G. Ruhe and C. Wohlin, Editors. 2014, Springer Berlin Heidelberg. p. 373-399.
4. Peixoto, D.C., G.R. Mateus, and R.F. Resende. *The Issues of Solving Staffing and Scheduling Problems in Software Development Projects*. in *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*. 2014. IEEE.
5. Chang, C.K., M.J. Christensen, and T. Zhang, *Genetic algorithms for project management.* Annals of Software Engineering, 2001. **11**(1): p. 107-139.
6. Alba, E. and J.F. Chicano, *Software project management with GAs.* Information Sciences, 2007. **177**(11): p. 2380-2401.
7. Kang, D., J. Jung, and D.H. Bae, *Constraint-based human resource allocation in software projects.* Software: Practice and Experience, 2011. **41**(5): p. 551-577.
8. Kwok, Y.-K. and I. Ahmad, *Benchmarking and comparison of the task graph scheduling algorithms.* Journal of Parallel and Distributed Computing, 1999. **59**(3): p. 381-422.
9. Kwok, Y.-K. and I. Ahmad, *Static scheduling algorithms for allocating directed task graphs to multiprocessors.* ACM Computing Surveys (CSUR), 1999. **31**(4): p. 406-471.
10. Antoniol, G., M. Di Penta, and M. Harman. *A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty*. in *Software Metrics, 2004. Proceedings. 10th International Symposium on*. 2004. IEEE.
11. Antoniol, G., M. Di Penta, and M. Harman. *Search-based techniques applied to optimization of project planning for a massive maintenance project*. in *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*. 2005. IE