GLOBAL ACADEMIC RESEARCH INSTITUTE

COLOMBO, SRI LANKA

International Journal of Engineering Science

ISSN 2424-645X

Volume: 01 | Issue: 01

On 30th June 2019

http://www.research.lk

Author: Dr. Rasika Aponsu

IIC University of Technology, Cambodia

GARI Publisher | Software Engineering | Volume: 01 | Issue: 01

Article ID: IN/GARI/JOU/2019/84 | Pages: 22-38 (17)

ISSN 2424-645X | Edit: GARI Editorial Team

Received: 22.05.2019 | Published: 30.06.2019

A METRIC-BASED APPROACH FOR MULTI-OBJECTIVE OVERTIME PLANNING IN SOFTWARE ENGINEERING PROJECTS

Dr. Rasika Aponsu

IIC University of Technology,

Cambodia

ABSTRACT

Software projects frequently experience unexpected overtime as a result of uncertainties and risks arising from evolving requirements and the pressure to meet the software product's time-to-market. This phenomenon induces stress among developers and may lead to substandard quality. The present study introduces a memetic algorithmic methodology to address the issue of overtime planning in software development projects. The issue is framed as a trivariate optimisation problem with the objective of minimising overtime hours, project makespan, and cost. The formulation effectively models the process of mistake production and dissemination over time through the use of simulation. The Multi-Objective Shuffled Frog-Leaping Algorithm (MOSFLA), which has been specifically developed for the purpose of overtime planning, is utilised to address the given problem. The researchers conducted empirical evaluation studies on six real-life software project datasets, employing three commonly utilised multi-objective quality indicators. The findings of the study indicate that MOSFLA exhibited superior performance compared to conventional overtime management systems in software engineering projects across all quality measures. Specifically, MOSFLA achieved values of 0.0118, 0.3893, and 0.0102 for Contribution (IC), Hypervolume (IHV), and Generational Distance (IGD), respectively. In all project situations, the proposed approach yielded superior outcomes in terms of IHV and IGD compared to the state-of-the-art approach (NSGA-IIV). Nevertheless, the proposed methodology exhibited superior performance compared to NSGA-IIV in around 67% of project instances in terms of IC.

Keywords: Human Resource Management, Software Engineering Projects, Metric-Based Approaches

INTRODUCTION

The field of Software Engineering focuses on addressing optimisation challenges in the development of software, aiming to enhance its speed, cost-effectiveness, reliability, scalability, responsiveness, maintainability, and testability [1]. Therefore, the successful completion of software engineering projects within the designated timeframe and budgetary constraints is contingent upon the implementation of effective software project management methodologies [2]. Software Project Management (SPM) encompasses a range of essential tasks, including cost estimation, project planning (including project scheduling and personnel), and quality management. These activities play a crucial role in determining the success of a software project [1]. These tasks frequently require the identification of a suitable equilibrium between objectives that are typically in competition and

conflict with one another. The SPM, or Software Project Management, is often conceptualised as a challenge including project scheduling and staffing [3]. This problem is typically addressed through the utilisation of the Search-Based Software Engineering (SBSE) technique. In the field of Search-Based Software Engineering (SBSE), a software engineering challenge is conceptualised as a search problem with the objective of identifying the most suitable solution that satisfies specific adequacy requirements. The objective is to reframe challenges in software engineering as optimisation problems based on search methods, and thereafter employ a range of meta-heuristic techniques to address these problems.

Software Project Scheduling (SPS) is an optimisation issue that aims to identify an optimal timetable for a software project. The objective is to ensure that both precedence and resource limitations are met, while minimising the overall project cost, which includes human salary and project duration [4]. The problem of SPS has been addressed by the utilisation of meta-heuristic optimisation techniques [3, 5-8]. Meta-heuristic algorithms are employed to explore a vast input space in pursuit of an optimal solution, with the guidance of a fitness function. The fitness function serves as a means to articulate the objectives and guide the investigation into potentially optimal regions within the search space. The algorithms employed in this context serve the purpose of automating and providing recommendations for software engineers, aiding them in making informed decisions during the planning phase of a software project. Nevertheless, it may not always be feasible to expect automated technologies to generate the initial project design. According to Ferrucci et al. (2019), their study findings indicated that practitioners expressed a preference for relying on their own judgements when creating the initial project plan. This preference stemmed from the fact that the allocation of staff to different work packages involves numerous decisions that are specific to the human and domain context. Consequently, an automated approach is deemed insufficiently equipped to effectively handle these decision-making processes. However, it is common for engineers to face the challenge of unguided initial project plans, which frequently result in developers having to work extra, often without prior planning, in order to fulfil project deadlines. There are multiple causes that can contribute to the necessity of working overtime. Several reasons contribute to the challenges faced in project management, including but not limited to: evolving and time-sensitive needs, shortened timelines, complexities in measuring project progress, and the imperative to align with the market's demand for timely delivery of new features. Nevertheless, the primary reasons that have been documented as contributing to unplanned overtime are a delayed alteration in requirements and a shortened time frame for product release [1, 10].

As anticipated, research has demonstrated that the allocation of excessive and unscheduled overtime has negative consequences on the well-being of developers and the quality of the software they generate. Nishikitani et al. [11] and Karita et al. [12] have reported an observation indicating a positive association between overtime work, long shifts, and stress indicators, such as sadness, rage, hostility, as well as an increase in equilibrium and motor-related issues among IT workers. In relation to the impact on software development, Akula and Cusick [13] discovered that extended periods of overtime labour have been shown to result in heightened levels of stress. Consequently, this heightened stress has been found to correlate with an increase in the number of defects observed

in software projects. The consequences of these impacts are indicative of substandard software, as developers persistently prioritise expeditious task completion over quality. The aforementioned fragments of evidence derived from scholarly sources emphasise the necessity of implementing a decision support methodology that can effectively manage overtime planning while considering factors such as staff productivity, cost, project duration, and product quality.

In order to address the adverse consequences of extended overtime in software development projects, engineers in the industry employ several strategies known as Overtime Management Strategies (OMS). In their study, Ferrucci et al. (2019) identified and discussed three prevalent overtime tactics employed by software engineers in the business. These strategies include Margarine (MAR) management, Critical Path Management (CPM), and Second Half (SH) management strategies. In the context of the MAR system, the allocation of overtime hours is distributed uniformly among all activities included in the schedule. In contrast, the Critical Path Method (CPM) incorporates additional hours into the schedule's critical path, but in the Schedule Half (SH) approach, overtime hours are allocated to the latter portion of the schedule to compensate for delays that occurred in the initial phase. However, there is a limited amount of research that has been conducted to effectively address the Overtime Planning Problem (OPP) using the SBSE approach. The present methodologies utilise genetic multi-objective evolutionary algorithms, specifically NSGA-II and its various adaptations [9, 10, 14]. As far as current information indicates, there has been no prior research conducted on the utilisation of alternative Multi-Objective Optimisation algorithms for the purpose of overtime planning in software projects. Hence, it is imperative to do further research on overtime planning procedures that have the potential to surpass existing overtime management tactics and state-of-the-art approaches.

The purpose of this study is to evaluate the efficacy of the memetic multi-objective evolutionary algorithm in addressing the issue of software project overtime planning. Specifically, the Multi-purpose Shuffled Frog Leaping Algorithm (MOSFLA) is employed as the primary approach for tackling this problem. In the context of optimisation problems, memetic algorithms have demonstrated superior efficiency and effectiveness compared to traditional evolutionary algorithms in certain problem domains [2, 15, 16]. Memetic algorithms can be conceptualised as a combination of a population-based global technique and a local search conducted by each individual within the population. Consequently, there is a growing recognition of memetic algorithms, notably in renowned combinatorial optimisation problems that have witnessed successful attainment of optimal solutions for huge instances, whereas alternative meta-heuristics have proven inadequate in achieving equivalent outcomes.

The subsequent section of this paper is structured in the following manner: Section 2 provides an overview of relevant literature pertaining to the scheduling of software projects over extended periods of time. Section 3 of the study was dedicated to the elucidation of the optimisation problem formulation. Section 4 of this study is dedicated to the proposed framework, which outlines the computational search technique utilised for addressing the optimisation problem (OPP). Additionally, this section discusses the evaluation metrics employed to assess the performance of the approach. Section 5 of the manuscript outlines the experimental configuration and provides an in-depth analysis of the obtained

results. The subsequent section, Section 6, serves as the concluding segment of the study.

RELATED WORKS

The impact of overtime on software quality was investigated by Akula and Cusick [13] through a statistical survey conducted on four real-life software projects. The recorded data for each project included the number of scheduled work hours, the actual overtime hours, and the number of flaws. The results of the statistical study indicated a positive correlation between the number of defects and the amount of extra hours in all of the projects that were examined. The study revealed a positive correlation between the number of defects and the extent of overtime, indicating that as extra hours grow, the defect count also tends to rise. Additionally, it was noted that prolonged overtime leads to increased stress levels among workers, which in turn negatively impacts the project's development. Despite the existing literature on the subject, there is a noticeable scarcity of research studies pertaining to the use of search-based meta-heuristic algorithms in the context of software project overtime planning. This investigation identified a limited number of three scholarly papers that have employed Search-Based Software Engineering (SBSE) to address the challenge at hand. The search-based optimisation approach to software overtime planning was introduced by Ferrucci et al. (9). The research paper introduced the conceptualization of software project overtime planning as a multi-objective optimisation problem. The aim of this study is to examine the impact of several overtime allocation choices on the project schedule. Each choice is designed to minimise project duration, overtime hours, and the risk of exceeding the project's allocated resources. This will be achieved by utilising three risk assessment models.

The computational search approach adopted in this study was NSGA-IIv, a variation of NSGA-II that incorporates a crossover operator specifically tailored for the overtime planning problem. In order to evaluate the efficacy of the aforementioned approach, the researchers conducted an empirical investigation including six authentic software projects. The experimental findings indicate that the proposed technique exhibits a statistically significant improvement compared to the conventional NSGA-II method in 76% of the conducted trials. Furthermore, the proposed methodology demonstrates superior performance when compared to the industry's prevailing overtime planning tactics. Nonetheless, the formulation employed by the authors neglected to consider the adverse impact of overtime on the quality of software, despite the existence of actual studies in the literature (references 11 and 13) that have demonstrated that the effort expended in rectifying the additional errors introduced by developers working overtime surpasses the gains in productivity.

In a recent study, Sarro et al. (2014) expanded upon the research conducted by Ferrucci et al. (2009) by introducing an adaptive approach for selecting meta-heuristic operators. The objective of this approach was to enhance the performance of algorithms within the context of the same problem formulation. The research introduced a novel iteration of NSGA-II, referred to as Adaptivevsc, that integrates the crossover operator proposed by Ferrucci et al. [9] with the adaptive genetic operators of NSGA-IIa, as proposed by Nebro et al. [17]. Furthermore, the research presented a novel approach for dynamically selecting genetic operators during the search process. The findings derived from empirical investigations conducted on eight authentic software

projects demonstrated that the suggested adaptive methodology exhibited superior performance compared to state-of-the-art methodologies in 93% of the trials. Furthermore, it was observed that the proposed technique consistently beat existing overtime planning strategies in all of the experiments, indicating a statistically significant improvement.

The approach presented by Ferrucci et al. [9] was further developed by Barros and Araujo [10] to incorporate the impact of overtime work on software quality. The research presented a novel approach to the OPP by incorporating simulation techniques to assess the impact of an augmented quantity of defects resulting from overtime on the project's length and expense. The primary aim of their research is to employ optimisation and simulation techniques in order to investigate the impact of various overtime planning rules on both the length and expense of the project. The computational search approach employed in this study was NSGA-II, which was utilised to optimise the planning process over time. The primary objective of this optimisation was to minimise the overall project length, project makespan, and cost. The methodology was contrasted with overtime tactics commonly employed in the industrial sector, as well as a comparable model that does not exhibit the detrimental impacts of overtime on product quality. The experimental findings demonstrated that NSGA-II exhibited superior performance compared to both the margarine and critical path overtime techniques. Nevertheless, the implementation of the Second-Half overtime management method yielded favourable outcomes when applied to NSGA-II within the specified framework. This discovery necessitates the development of a more effective search methodology that has the potential to surpass all existing techniques in terms of performance within the same framework.

This study closely aligns with the research conducted by Barros and Araujo [10] in terms of problem formulation. However, it diverges in terms of the computational search strategy employed, utilising a memetic algorithm known as MOSFLA.

Problem Formulation

The present work utilises a decision problem model proposed by Barros and Araujo [10] to address the issue of overtime planning. This model incorporates the complexities associated with error production and propagation resulting from overtime, employing simulation techniques. This methodology calculates the potential influence of escalated defect counts resulting from overtime on the overall length and expense of the project.

Consider a project schedule that may be described as a Directed Acyclic Graph (DAG). This DAG consists of a set of nodes, denoted as WP = {wp1, wp2, ..., wpn}, which represent the Work Packages involved in the project. Additionally, there is a set of edges, denoted as DP = {(wpi, wpj): $1 \leq i \leq n$, $1 \leq j \leq n$}, which represents the dependencies among the Work Packages.Every work package (WP) is distinguished by the level of effort needed to finish it, which is quantified in terms of function points (FP). Each dependency in the Dependency Structure Matrix (DSM) signifies that the initiation of the development of a certain work package (wpj) is contingent upon the conclusion of the analysis phase of another work package (wpi).

Based on the empirical estimation of average productivity of 27.8 FP/developer-month for Information Systems (IS) projects [18], the expected length of a work package may be determined by considering the size of the work package in FP.

The equation (1) represents the duration (D) of an event, which is a function of the effort (wpi) divided by a constant value of 27.8.

The project's minimum duration is determined by the Critical Path, which refers to the longest path in terms of duration. The utilisation of the critical route methodology has been a longstanding practise in the construction of software project schedules for several decades. Nevertheless, the primary concern lies in the analysis of the impacts of overtime assignments on the schedule, with the objective of minimising project makespan, expense, and the use of overtime. In the context of error propagation within software development projects, Jones (18) discovered that developers, on average, introduced four errors per functional point (FP) during the analysis, design, and coding phases of a component. This phenomenon has an impact on the length of testing procedures. In regards to mistake production resulting from overtime, Abdel-Hamid and Madnick [19] established a model to examine the correlation between the extent of overtime invested by developers and the quantity of faults introduced into the software they generate. According to this model, developers who work 10 hours per day are estimated to introduce 20% more errors compared to those who work regular shifts of 8 hours per day. Additionally, developers working 12 hours per day are estimated to introduce 50% more errors. The complex link between overtime and error generation rates, along with the dynamics of error transmission, gives rise to a compound model that necessitates the use of an optimisation technique driven by simulation. Hence, this research integrates optimisation and simulation techniques to effectively address the issue of software overtime planning optimisation.

Optimization Objectives

The OPP formulation in this study tries to minimize:

(3)

• Amount of Overtime Hours (OH) = $\sum n\ overtime\ (wp)$ (2)

i=1

• Project Make Span (PMS) = $\sum wp \in CP\ duration\ (wpi)$

i

• Project Cost (PC) = $\sum n\ C\ (wp)$ + $C\ (wp)$ (4) i=1$r\ i\ o\ i$

Subject to the constraints:

0 ≤ $overtime(wpi)$ ≤ max$overtime(wpi)$, 1 ≤ i ≤ n

DPviolation< 1.

The variable "Overtime Hours (OH)" represents the cumulative number of hours spent on activities beyond the regular working hours, as determined by the computational search technique. The Project Makespan (PMS) is determined as the longest path in the graph that adheres to precedence constraints. It is computed by summing the durations of the activities along this path. The Critical Path (CP) refers to the sequence of activities in a project that determines the overall duration of the project. The duration of each activity is denoted as wp_i. The computation of the time of an activity is derived from the simulation, since it incorporates the integration of error production and propagation dynamics into the schedule of the project. The Project Cost (PC) is determined by aggregating the costs associated with each activity. These costs are contingent upon the number of regular and overtime hours expended by the developer in order to achieve the anticipated results. The cost law proposed by Barros and Araujo [10] in Brazil is employed for the purpose of representing the cost associated with a particular activity. It may be inferred that in the given scenario, the cost of a regular working hour is denoted as C.

Furthermore, each of the initial two overtime hours incurs a cost of 120% of C, while each subsequent two hours of overtime incur a cost of 150% of C. The variable C_r in objective (iii) represents the cost of regular working hours, while C_o represents the cost of overtime hours.

The possible solution for the Overtime Project Problem (OPP) is represented as a collection of integer values that indicate the amount of additional hours to be allocated per day for each individual Work Package (WP) in the project schedule. The representation consists of a linear array of numbers, where each integer corresponds to the number of overtime hours allocated for a specific activity indexed in the array. It is vital to acknowledge that the duration of standard working hours and the maximum allowable extra hours are contingent upon the specific country's regulations and policies. The study conducted by Barros and Araujo [10] establishes a standard working schedule of 8 hours each day, with a maximum allowance of 4 hours for overtime.

PROPOSED FRAMEWORK

This part outlines the computational search technique utilised for solving the overtime planning problem as described in part 3. Additionally, it introduces the evaluation metrics that will be employed to assess the effectiveness of the suggested computational approach.

Computational Search Approach

In this study, we utilize a computational search method known as the Multi-Objective Shuffled Frog-Leaping Algorithm (MOSFLA). This algorithm is built on the framework of the original SFLA algorithm published by Eusuff and Lansey [20]. Additionally, we incorporate an archiving strategy that is based on the adaptive niche methodology proposed by Cui [21]. The niche strategy is employed as a means of preserving the non-dominated solutions. The algorithm enhances the population sorting technique and memetic evolution process in order to accommodate Multi-Objective Optimisation (MOO).

As a result of the parallel evolution method employed by the algorithm, the solutions undergo evolutionary changes in divergent directions. This characteristic renders MOSFLA particularly suitable for addressing MOO challenges, such as the one under investigation in this paper. The MOSFLA method is depicted in Figure 1, as modified by Yinghai et al. [22]. The use of MOSFLA has demonstrated good outcomes across several domains, including typical optimisation test problem cases [23], the reservoir flood management problem [22], and the robot path planning problem [24].

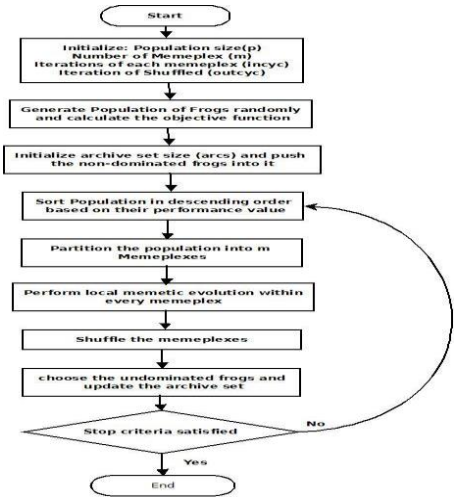

Fig. 1. Flowchart of MOSFLA [22].

MOSFLA Design for Overtime Planning Problem

In order to address the overtime planning problem, the method for adapting the MOSFLA is formulated as follows:

• Population creation and sorting strategy

Due to the stochastic nature of MOSFLA, the population of viable solutions, referred to as "frogs," is generated in a random manner. The determination of whether a solution is accepted or rejected is contingent upon its ability to satisfy the maximum overtime limits. In the SFLA (Shuffled Frog Leaping Algorithm), frogs are arranged in a descending order based on their performance values, which are determined by a fitness function. In the context of single-objective problems, it is customary to establish the performance value by directly assigning it as the value of the objective function. The application of this approach is not feasible for multi-objective problems. Numerous methodologies have been employed in scholarly works to assess the performance value in the context of multi-objective problems.

Several methodologies have been employed in this study, including the utilisation of the Pareto dominance relation to evaluate the solutions. Additionally, the crowding distance of individual solutions has been considered. Furthermore, a combination of the crowding distance inside the non-dominated solutions and the hamming distance between dominated and non-dominated solutions in the population has been investigated [22, 23, 25]. The present study utilises a novel hybrid multi-objective fitness function that incorporates both the crowding distance of individual frogs and the rank gained from sorting Pareto fronts. This fitness function is applied to evaluate the fitness of each solution within the population.

To be more precise, the rank is determined through the implementation of the subsequent procedure:

In the first step, the non-dominated solutions present in the initial population are assigned to the first rank, also known as rank 0, and subsequently eliminated.

In the second step, the subsequent non-dominated solutions are identified from the remaining solutions and incorporated into the subsequent rank, which is designated as rank 1. These solutions are also removed in accordance with the appropriate measures.

The aforementioned technique is iterated until the population no longer contains any additional solutions.

Following the completion of the ranking process, the computation of the crowding distance (Cd) is conducted for all solutions within the same rank. Assuming that the multi-objective optimisation (MOO) entails a set of predefined goals, the crowding distance of each frog is computed as follows:

$$C = \sum r\ |P[i+1] \cdot f - P[i-1] \cdot f| \quad (5)\ d\ k=1\ k\ k$$

C^d is the crowding distance of the i^{th} frog in the rank set, P[i+ 1] fk and P[i- 1] fk are k^{th} objective function values of two adjacent frogs.

The total multi-objective fitness function presented by Alejandro et al. (24) is computed using the following formula. The term is characterised as:

$$MOFit = {}^{1} \quad (6)\ 2rank + {}^{1}$$
$$1 + C_d$$

Memplex formation

The sorted frogs are stored in an array X = {P(i), i = 1, . . . ,n}. X is then partitioned into m memeplexes, i.e., $Y^1, Y^2, \ldots, Y^m$, each containing n frogs, such that:

$Y^k = [P(i)^k | P(i)^k = P[k + m(i-1)]],\ i = 1,..n\ k = 1,..m$ (7)

In this way, for m = 4, frog in position 1 goes to memeplex 1, position 2 to memeplex 2, position 3 to memeplex 3, position 4 to memeplex 4. Then frog in position 5 goes to memeplex 1, and so on.

Memetic evolution in memeplex

Within every memeplex, there exists a process of memetic evolution wherein virtual frogs undergo improvement through the transfer and sharing of memes, with the aim of enhancing the performance of the least successful frog. In the initial implementation of the Structured Fitness Landscape Algorithm (SFLA), the evolutionary process involves enhancing the fitness of the least optimal frog inside each memeplex.

$d=rand*(x_b - x_w)$, $newxw=oldx_w + d$ (8)

where x_b = local best frog in the memeplex and xw = worst frog in the memeplex.

The current evolutionary progression exhibits inefficiency due to its restriction of the new frog's habitat to the region bounded by xw and xb. This measure solely reinforces an enhancement in the local bound. To expand the evolutionary scope of newxw beyond its local confines, the researchers Yinghai et al. [22] have suggested implementing the following evolutionary step.

$d=2*rand*(x_b - x_w)$, $newx_w = oldx_w + d$ (9)

This step increases the evolutionary space by a factor of two, allowing the position of newxw to reach a value of 2(xb−xw). Fundamentally, the process of evolution has the capability to generate solutions that surpass the existing optimal solution within every memplex.

To achieve local evolution, the first frog (yk[1]) in the kth memeplex is designated as the local best frog (xb), whereas the last frog (yk[n]) is designated as the worst frog (xw). In order for frogs to undergo evolutionary progress towards Pareto optimality, the global best frog (xg) is designated as a solution that is randomly selected from the current archive set. Additionally, the location of the least optimal frog is modified using Equation (9). The calculation of objective function values is performed, followed by a comparison of the Pareto dominance connection between newxw and oldxw.

• If $newx_w$ dominates $oldx_w$, then y^k[n] is replaced with newxw.

• If $oldx_w$ dominates $newx_w$, then go to iii;

• Step two is recomputed by substituting xb with xg in Eq. (9).

• If $newx_w$ dominates the $oldx_w$, then y^k[n] is replaced with newxw and

• If $oldx_w$ dominates $newx_w$, then go to iv;

A novel solution is developed by a random process to replace the frog with the lowest fitness value. In order to steer this process towards an evolutionary trajectory, the novel solution is derived through the random generation of a new frog within the vicinity of the globally optimal frog, denoted as xg.

After the process of memetic evolution, the memeplex Yk undergoes an update and reorganisation.

Each memeplex undergoes repeats of steps i to v for a predetermined number of times.

The topic of discussion pertains to the approach of shuffling and archiving.

Following a series of memetic evolutions, the memeplexes undergo a process of consolidation and arrangement, wherein they are organised in a descending order based on their MOfit value. The process of identifying non-dominated solutions is conducted, and subsequently, these solutions are incorporated into the archive set. The utilisation of the archiving method is prevalent in numerous MOO algorithms as a means of preserving the collection of non-dominated solutions. The utilisation of niche methods is a highly successful strategy for promoting and maintaining

variety among the non-dominated solutions inside a given set. The niche-based archiving method use the niche radius as a means to determine the sharing fitness of non-dominated solutions. The quantification of fitness sharing is determined using the following formula:

$F(i) = 1/\sum^{q} sh(\ d\ )\ j{=}1\ ij$
where:
$1{-}(\ ^{dij}\ )^{\alpha}\ sh(dij) = \{\ ^{\sigma}share$
0
$d_{ij} < {}^{\sigma}share$
$d_{ij} > {}^{\sigma}share$
(10)
(11)

The sharing fitness of the ith non-dominated solution is denoted as F(i). The variable q represents the number of solutions in the archive set. The sharing function between the ith and jth non-dominated solutions is represented as sh(dij), where dij is the Euclidean distance in the objective space between the ith and jth non-dominated solutions. The constant coefficient α and the niche radius σshare are also included in the equation.

The relationship between the niche radius and F(i) is evident. An improperly determined niche radius can result in a non-uniform distribution of the non-dominated solutions. This study utilises a self-adaptive calculation method proposed by Chui [21] to determine the niche radius ($\sigma share$) based on the number and distribution of solutions in the archive set. The difficulty of specifying the niche radius a priori is taken into consideration, and therefore, the method automatically computes and adjusts it during the iteration procedure. The computation approach is provided in equations (12) and (13).

C
$^{\sigma}share = \{\sum^{q} i{=}1$
$if\ q < 2$
$d_{i\sqrt{q}}$ (12) $if\ q \geq 2$
$d = min\ (\ \|\ F\ (x) - iij$
$F\ (x)\ \|\ )$ for i, j = 1,2,...,q such that j ≠I (13)

Let q symbolise the quantity of solutions within the archive set. The variable di represents the minimal Euclidean distance in the objective space between the ith non-dominated solution and the remaining solutions. Additionally, C is a positive constant that is often assigned a value of 1. The niche radius is determined by computing the average value of di for all non-dominated solutions inside the archive set.

Multi-objective evaluation measures used

This research utilises three quantitative metrics of solution set quality: Contributions (IC), Hypervolume (IHV), and Generational Distance (IGD), which were previously employed by Ferrucci et al. [9] and Barros and Araujo [10]. The values are quantified within the interval [0, 1].

In computational science, the indicator of convergence (IC) refers to a quantitative metric used to assess the degree to which an algorithm, denoted as A, generates solutions that reside on the reference front RS. The calculation involves determining the ratio of solutions in RS generated by A [26]. This ratio is formally defined as follows:

Let C represent the Pareto solution set that is shared by both A and RS. W represents the solution sets in A that dominate other solutions in RS, and N represents the set of solutions in A that have a dominance relation with no solution in RS. An optimal Pareto front should exhibit a high IC (Indicator Contribution) value and make a significant contribution to the reference front.

The Inverted Hypervolume (IHV) is a metric used to determine the volume

enclosed within the objective space by the set of non-dominated solutions generated by a specific algorithm. This metric combines elements of both convergence and diversity. The study utilises the Optimal 3D Hypervolume Algorithm as reported by Paquete et al. (27). The algorithm traverses a front that has been sorted based on a single aim, while simultaneously keeping track of the entire two-dimensional area encompassing the points that have been examined thus far.

The position of each point, denoted as p, in the front is determined by querying a height-balanced binary tree with the remaining objectives. If a variable p is determined to be dominated, it is subsequently eliminated. In the event that a point p exhibits dominance over other points, those points are subsequently removed from the tree structure. If necessary, the two-dimensional area is subsequently updated in constant time. The vertical distance between point p and the subsequent lower point, referred to as the slice depth, is thereafter multiplied by the area and subsequently incorporated into the total volume. The algorithm's pseudo-code is depicted in Figure 2. The algorithm's performance improves as the value of IHV increases.

The Index of Generational Distance (IGD) is a metric used to assess the convergence of an algorithm by quantifying the average distance between the solution set S generated by the algorithm and the reference set RS. The calculation of the distance between set S and set RS in an N-dimensional objective space involves determining the average Euclidean distance across N dimensions. This value is obtained by measuring the distance between each point in set S and its closest neighbouring point in set RS. The term is defined in the following manner [28].

$$IGD(S) = \frac{1}{|RS|} \sum_{y \in RS} min\{d_{xy} |_x \in S\} \quad (15)$$

The d_{xy} isthedistancebetweenasolutionxinSandareferencesolutionyin RS in the N-dimensional objective space as defined in Eq. (16):

$$d = \sqrt{(f(y) - f(x))^2 + \cdots + (f(y) - f(x))^2} \quad (16)$$

$xy11\ NN$

where f (x) is the i^{th} objective function values of a solution x. Good fronts possess i

low IGD and thus, are closer to the reference front.

Initialize tree, sort PS in 3rd objective and set Volume to 0 Set p = head (PS), ps = tail (PS), area = p[0] * p[1], z = p For each p in PS

Search tree for point q to the right of p

If p is not dominated

Increase volume by slice between z and p

z=p

For each point s in tree dominated by p Remove s from tree

RESULTS AND DISCUSSION

This section presents a comprehensive analysis of the design of our empirical experimental investigation, encompassing the dataset employed, parameter configurations, and the outcomes of the experiments.

Software project data used

This study utilises six (6) sets of real-life software project data that were collected by Barros and Araujo [10] and have been made publically accessible for the purpose of replication and validation in research. The dataset has been summarised and presented in Table 1. The OMET programme is a software tool designed for the purpose of effectively managing meteorological data. The WAMS system is an air traffic routing control system that effectively manages traffic control communications. The Profile and Configuration Settings Management (PARM) system is

responsible for the storage and management of user profiles and their associated configuration settings, which are utilised by a multitude of applications.

The Personnel System for Online Authentication (PSOA) is a comprehensive system designed to effectively handle the authentication and authorization of users within enterprise systems. ACAD is a comprehensive academic portal system designed to effectively handle the records of university students and staff members. The WMET system is responsible for the management and storage of meteorological data within a database.

Parameter setting

The experimental setting of three major factors was conducted in response to the non-deterministic characteristics of MOSFLA. The parameters were subjected to variation in successive iterations, and the most optimal values were chosen for the real experiment. The shuffling iteration, which is responsible for determining the stopping criterion, is evaluated using the values 500, 1000, 1500, and 2000. To account for the inherent randomness of MOSFLA, the optimisation process was performed 30 times for each value and instance. Subsequently, a reference front was constructed using the non-dominated solutions gathered from all 30 cycles for each instance. Next, the Inverted Generational Distance (IGD) was calculated for the offspring produced in each iteration in order to identify the optimal and efficient value.

To determine the optimal beginning population size, a comparable methodology was employed. A comparison was made between the population sizes of 2m, 3m, and 4m, where 'm' represents the number of actions for each instance. Table 2 presents the initial findings of the conducted experiment on the specific case of instance ACAD. In all experimental trials, the quantity of memeplexes was established at 5. The process for the number of iterations of memetic evolution within each memeplex was consistently applied for each case. The value of the memeplex is determined by the quantity of frogs, denoted as 'n'. Specifically, the value is established as 2n, 4n, and 8n. The preliminary findings of the experiment conducted on instance ACAD are presented in Table 3. Based on the first findings, it can be deduced that the combination of shuffling iteration 1500, population size 4m (where m represents the number of activities in the given instance), and evolution iteration value 4n (where n represents the number of frogs in each memeplex) yielded the most favourable outcomes across all the examined combinations. Hence, the three primary parameters have been established appropriately for the empirical investigation.

Experimental results

The Java implementation of MOSFLA was utilised for all project instances, with the parameters configured as previously stated in the preceding subsection. To account for the inherent variability of the algorithm, each experiment is repeated 30 times and the outcomes are subsequently averaged. Each instance is evaluated using the multi-objective quality indicators Contributions (IC), Hypervolume (IHV), and Generational Distance (IGD). The reference front is constructed by aggregating all the fronts generated from each individual run. The findings of the IC, IHV, and IGD metrics for all occurrences under consideration are presented in Table 4.

The results presented in Table 4 demonstrate the strong performance of the proposed memetic algorithm. Specifically, the algorithm achieved a high hypervolume value of around 0.7, indicating its success in exploring the solution space. Additionally, the method

exhibited a high contribution value of approximately 0.4, further highlighting its efficacy in improving the quality of solutions. Furthermore, the Generational distance was found to be quite low, suggesting that the system is robust and capable of generating solutions that are close to the true Pareto front. Overall, these findings underscore the effectiveness and robustness of the suggested memetic algorithm. It can be inferred that the algorithm exhibited superior performance in the context of large-scale projects, as evidenced by its ability to generate the highest hypervolume and generational distance values for instances with a substantial number of activities (108 in the case of PARM). However, it is worth noting that the algorithm achieved its optimal outcome in terms of Contribution in a medium-scale project with 84 activities (OMET).

Based on the analysis, it can be inferred that MOSFLA has the highest level of suitability for addressing the overtime planning problem seen in large-scale software engineering projects. In order to conduct a comprehensive assessment of the algorithm's efficacy, a comparative analysis was conducted, pitting its performance against conventional overtime management tactics commonly employed in the software industry. In this study, we evaluate the OPP formulation utilising the MOSFLA search method with three OMS strategies proposed by Ferruci et al. [9], namely "margarine" (MAR), Critical Path (CPM), and Second Half (SH). The fronts that were generated by MOSFLA throughout 30 optimisation cycles were compared to the front generated by each OMS based on the quality metrics. The reference front was constructed using the Pareto fronts derived from all the optimisation multi-objective strategies (OMS) and the multi-objective shuffled frog leaping algorithm (MOSFLA). The findings of the operational management strategies are derived from the source referenced as [10]. The outcomes of MOSFLA in comparison to all other OMS techniques are presented in Table 5.

In terms of the Contributions (IC) quality indicator, it is evident that MOSFLA has superior performance compared to all other overtime management techniques across all six cases, with the exception of the ACAD instance where it achieved the lowest value of 0.0016. Figure 3 presents a comparative analysis of the outcomes derived from the Contribution (IC) indicator. In terms of the Hypervolume (IHV) quality indicator, it is evident that MOSFLA outperforms all other overtime management methods across all instances, with the exception of the ACAD instance. In the ACAD instance, SH achieved the highest value of 0.3589, while MOSLA closely followed with a value of 0.3402. The information is clearly depicted in Figure 4.

The potential reason for the suboptimal performance of MOSFLA in the ACAD instance could be attributed to the relatively small scale of the project. It is worth noting that the algorithm tends to exhibit improved performance when used to larger projects, as evidenced by the findings presented in Table 4. When evaluating Generational Distance (IGD), it is evident that MOSFLA consistently surpasses all other overtime management solutions, demonstrating the lowest values across all six project cases.

Figure 5 illustrates the Generational Distances (IGD) associated with both MOSFLA and overtime management techniques. The MOSFLA algorithm consistently yields superior results in terms of the Inverted Generational Distance (IGD) and Inverted Hypervolume (IHV) metrics across all instances. The IC MOSFLA demonstrated

marginal improvements in comparison to the alternative OMS.

Table 6 presents the average values of Contribution (IC), Hypervolume (IHV), and Generational Distance (IGD) obtained by MOSFLA and other OMS techniques, ensuring a proper comparison.

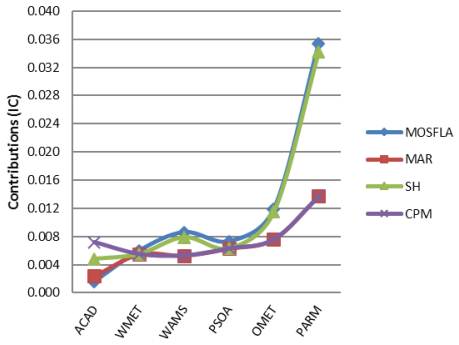

Fig. 3. Contributions of MOSFLA and other OMS

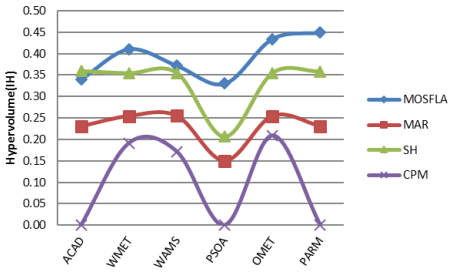

Fig. 4. Hypervolumes of MOSFLA and other OMS

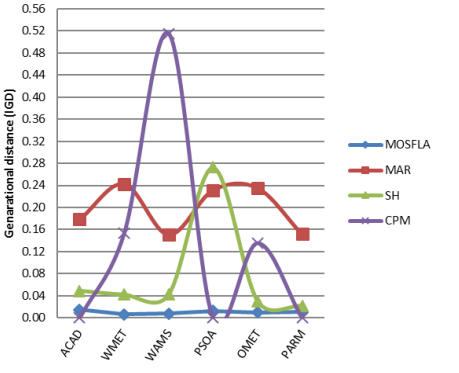

Fig. 5. Generational Distances of MOSFLA and other OMS

The performance of the proposed algorithm MOSFLA, as formulated in the current study, exhibited superior results compared to all other techniques employed over time. On average, the Contribution (IC) quality indicator exhibited the greatest values of 0.0118, while the Hypervolume (IHV) quality indicator demonstrated the highest value of 0.389. Conversely, the Generational Distance (IGD) quality indicator had the lowest value of 0.0102. On average, each run of MOSFLA made a contribution of 5.8 solutions to the reference front, while OMS made an overall contribution of 6.83 solutions. This finding suggests that the proposed approach accounted for 46% of all generated solutions, which is only slightly lower than the total number of solutions produced by the OMS strategies. This demonstrates the overall superiority of the MOSFLA algorithm compared to the currently practised overtime management strategies in industries.

Finally, for the proposed memetic strategy to be considered, it is imperative that it demonstrates superior performance compared to the current state-of-the-art technique for the given problem. The present solution approaches in Search-Based Software Engineering (SBSE) for addressing the issue of planning software projects' overtime primarily utilise NSGA-II and its variants [9, 10, 14]. In order to assess the performance of MOSFLA, we compare its results with those obtained by Barros and Araujo [10] using NSGA-IIV, as our study is built around the same dataset. The findings of the comparison are presented in Table 7. In terms of Contribution (Ic), it is evident that NSGA-IIv yielded greater values of 0.2636 and 0.0848 in the ACAD and WMET cases, respectively. Nevertheless, when the project instances become larger, MOSFLA outperforms NSGA-IIv in terms of performance value. Specifically, MOSFLA exhibits greater values of Contribution (Ic) in WAMS, PSOA,

OMET, and PARM, with the margin rising in direct proportion.

The data suggests that there is a positive correlation between project size and the performance of MOSFLA, as the former increases, the latter also increases. Conversely, NSGA-IIv exhibits a negative correlation with project size, as its performance drops as projects grow larger. The graphical representation of the observed change may be seen in Figure 6. The aforementioned outcome highlights the efficacy of MOSFLA in addressing issues of significant magnitude. In all project instances, MOSFLA exhibited superior performance compared to NSGA-IIv, as evidenced by its higher Hypervolume (IHV) values and lower Generational Distance (IGD) values. Notably, MOSFLA achieved the greatest IHV of 0.4489 in the PARM project instance, while also attaining the lowest IGD value of 0.0065 in the WMET project instance. The graphical representation of the Inverted Hypervolume (IHV) of MOFLA and NSGA-IIv is depicted in Figure 7.

CONCLUSION

A novel memetic method, utilising the Multi-Objective Shuffled Frog Leaping method (MOSFLA), has been devised for the purpose of multi-objective overtime planning in software engineering projects. The Overtime Planning Problem is expressed as a three-objective optimisation problem that encompasses the dynamics of error creation and propagation resulting from the implementation of overtime, utilising simulation techniques. A modified version of the Multi-Objective Simulated Flight Algorithm (MOSFLA), known as HyperVolume (IHV) and Contribution (IC), has been specifically developed for the purpose of addressing the overtime planning problem at hand. The method utilises a self-adaptive niche-based archiving strategy in order to preserve the non-dominated solution. The algorithm was adapted to MOO by the implementation of efficient sorting and memetic evolution techniques. The algorithm's effectiveness was assessed by empirical evaluation using a dataset derived from real-life software projects. The results indicate that the technique is highly effective in the management of medium and large-scale software development. Notably, it surpassed all existing overtime management strategies across many quality measures. The memetic technique demonstrates superior performance compared to the state-of-the-art approach (NSGA-IIv) across all quality indicators. In subsequent research endeavours, we want to conduct a comprehensive examination of the effect magnitude and statistical significance of the findings through the utilisation of inferential statistical techniques. Additionally, our research aims to empirically assess the impact of overtime on the programme's quality by employing software quality prediction tools [29-31].

Nomenclatures

C_d Crowding distance

C_o Cost of overtime hours

C_r Cost of regular hours

d_{ij} Euclidean distance of objective space between the i^{th} and

j^{th} non-dominated solutions F(i) Sharing Fitness

IC Contribution

IGD Generational Distance

IHV Hypervolume

MOFit Multiobjective Fitness Function

$Sh(d_{ij})$ Sharing function of i^{th} and j^{th} non-dominated solutions xb Local best frog

x_g Global best frog

x_w Worst frog

Y^k K^{th} memeplex

Greek Symbols

α Sharing constant-coefficient σ_{share} Niche radius

Abbreviations

CPM Critical Path Management DAG Directed Acyclic Graph DP DePendency

FP Function Points

MAR MARgarine Management

MOO Multi-Objective Optimization

MOSFLA Multi-Objective Shuffled Frog-Leaping Algorithm NSGA-II Non-dominated Sorting Genetic Algorithm II

OH Overtime Hours

OMS Overtime Management Strategies

OPP Overtime Planning Problem

REFERENCES

Oladele, R.O.; and Mojeed, H.A. (2014). A shuffled frog-leaping algorithm for optimal software project planning. African Journal of Computing and ICTs, 7(1), 147-152.

Ren, J. (2013). Search based software project management. Ph.D. Thesis. University College London, London, United Kingdom.

Patil, N.; Sawanti, K.; Warade, P.; and Shinde, Y. (2014). Survey paper for software project scheduling and staffing problem. International Journal of Advanced Research in Computer and Communication Engineering, 3(1), 215-324.

Chang, C.; Christensen, M.; and Zhang, T. (2001). Genetic algorithms for project management. Annals of Software Engineering, 11(1), 107 - 139.

Alba, E.; and Chicano, F. (2007). Software project management with GA's. Information Sciences, 177(11), 2380-2401.

Gueorguiev, S.; Harman, M.; and Antoniol, G. (2009). Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering. Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO). Montral, Canada, 1673-1680.

Stylianou, C.; and Andreou, A.S. (2013.) A multi-objective genetic algorithm for intelligent software project scheduling and team staffing. Intelligent Decision Technologies, 7(1), 59-80.

Ferrucci, F.; Harman, M.; Ren, J.; and Sarro, F. (2013). Not going to take this anymore: Multi-objective overtime planning for software engineering projects. Proceedings of the International Conference on Software Engineering (ICSE). Piscataway, New Jersey, United States of America, 462-471.

Barros, M.d.O.; and Araujo, L.A.O.d.J. (2016). Learning overtime dynamics through multiobjective optimization. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). Denver, Colorado, United States of America, 1061-1068.

Nishikitani, M.; Nakao, M.; Karita, K.; Nomura, K.; and Yano, E. (2005). Influence of overtime work, sleep duration, and perceived job characteristics on the physical and mental status of software engineers. Industrial Health, 43(4), 623-629.

Karita,K.;Nakao,M.;Nishikitani,M.;Iwata,T.; Murata,K.;andYano,E.(2006). Effect of overtime work and insufficient sleep on postural sway information-technology workers. Journal of Occupational Health, 48(1), 65-68.

Sarro,V.;Ferrucci,F.;Harman,M.;Mannay,A.; andRen,J.(2017).Adaptive multi-objective evolutionary algorithms for overtime planning in software projects. IEEE Transactions on Software Engineering, 43(10), 898-917.

Deng, J.; and Wang, L. (2017). A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem. Swarm and Evolutionary Computation, 32, 121-131.

Poonam, G. (2009). A comparison between memetic algorithm and genetic

algorithm for the cryptanalysis of simplified data encryption standard algorithm. International Journal of Network Security and its Applications (IJNSA), 1(1), 34-42.

Nebro, A.J.; Durillo, J.J.; Machin, M.; Coello, C.A.C.; and Dorronsoro, B. (2013). A study of the combination of variation operators in the NSGA-II algorithm. Lecture Notes in Computer Science, 8109.

Jones, C. (2000). Software assessments, benchmarks, and best practices. Boston, Massachusetts, United States of America: Addison-Wesley Longman Publishing Co.

Abdel-Hamid, T.; and Madnick, S.E. (1991). Software project dynamics: An integrated approach. Upper Saddle River, New Jersey, United States of America: Prentice-Hall.

Eusuff,M.;andLansey,K.(2003).Optimizationof waterdistributionnetwork design using the shuffled frog leaping algorithm. Journal of Water Resources. Planning and Management, 129(3), 210-225.

Cui, X.X. (2006). Multi-objective evolutionary algorithms and their applications. Beijing, China: National Defence Industry Press.

Alejandro, H.; Miguel, A.V.; Joaquín, F.; and Nieves, P. (2015). MOSFLA- MRPP: Multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning. Engineering Applications of Artificial Intelligence, 44, 123-136.

Rahimi-Vahed, A.; and Mirzaei, A.H. (2007). A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem. Computer & Industrial Engineering, 53(4), 642-666.

Elena, S.N. (2007). Performance measures for multi-objective optimization algorithms. Matematică - Informatică - Fizică, 109(1), 19-28.