VFAST Transactions on Software Engineering
https://vfast.org/journals/index.php/VTSE@ 2025, ISSN(e): 2309-3978, ISSN(p): 2411-6246
Volume 13, Number 3, July-September 2025 *pp:158-175*

Solving Software Project Scheduling Challenges Using a Search-Based Approach

Mahnoor Iftekhar[1], **Syed Sarmad Ali**[2], **Muhammad Hussain Mughal**[3*]

[1]Department of Software Engineering, Sir Syed University of Engineering and Technology (SSUET), Karachi; [2]Department of Computer Science and Engineering, Beihang University, (BUAA), P.R.China.; [3]Department of Computer Science, Sukkur IBA University, Pakistan

Keywords: *Project scheduling, Metaheuristic Algorithms, Software Project Scheduling Problem, Machine Learning, Genetic Algorithms.*

Journal Info:
Submitted:
July 05, 2025
Accepted:
September 16, 2025
Published:
September 24, 2025

Abstract

Efficient scheduling is a cornerstone of successful software project management, directly influencing delivery timelines, cost control, and resource utilization. However, inadequate scheduling methodologies remain a critical factor behind project delays and budget overruns, particularly in contexts requiring rapid deployment such as the COVID-19 pandemic. This study addresses the Software Project Scheduling Problem (SPSP) through a search-based approach leveraging metaheuristic optimization techniques. Specifically, we investigate the effectiveness of Genetic Algorithms (GA), Tabu Search (TS), and their hybridization, in comparison with alternative metaheuristics such as the Firefly and Dragonfly algorithms. The International Software Benchmarking Standards Group (ISBSG) dataset, comprising over 2,000 global software projects, is employed as the empirical basis for validation. The research design encompasses a comprehensive literature review, formulation of research questions, and systematic application of GA and GA–TS hybrid models. Experimental evaluation reveals that the hybrid approach achieves substantial improvements over standalone GA, with mean fitness values increasing by approximately 40% across 100 iterations (from 0.948 to 1.887 in the final 10 iterations). Furthermore, the hybrid model reduced convergence time by nearly 30%, enhanced resource allocation accuracy, improved project duration estimates by 25%, and lowered projected costs by 20%. These results demonstrate that the GA–TS hybrid consistently provides more robust, efficient, and reliable scheduling solutions. The findings contribute to both theory and practice by validating the superiority of hybrid metaheuristics in addressing the inherent complexity and nonlinearity of SPSP. They further highlight the potential of search-based techniques to improve real-world project management outcomes in dynamic and resource-constrained environments. Future research should extend this work by integrating additional metaheuristic combinations, incorporating uncertainty modeling, and testing across diverse datasets to generalize applicability and strengthen practical adoption.

***Correspondence author email address:** muhammad.hussain@iba-suk.edu.pk
DOI: 10.21015/vtse.v13i3.2186

1 Introduction

Inefficient scheduling practices not only result in significant project delays and cost overruns but also hinder the effective allocation of scarce resources, thereby exacerbating risks in already volatile project environments. These challenges highlight the pressing need for advanced computational approaches capable of addressing the inherent complexity and uncertainty of software project scheduling.

The concept of *metaheuristics* offers a promising paradigm for addressing such optimization challenges. A metaheuristic can be broadly defined as a high-level, problem-independent strategy that guides lower-level heuristics to efficiently explore the solution space and generate near-optimal solutions. The term was first introduced by Glover in the late 1980s, although metaheuristic strategies had been employed in various problem domains well before their formalization. Unlike traditional heuristics, which are tailored to specific problem instances, metaheuristics provide generalized frameworks that balance exploration and exploitation to navigate large and complex search spaces. Notable examples include Genetic Algorithms (GA) [1], Firefly Algorithm [2], Dragonfly Algorithm [3], Pareto Archived Evolution Strategy (PAES) [4], and Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [5]. Each of these algorithms has been successfully applied in optimization tasks, and their adaptability makes them particularly suitable for addressing the Software Project Scheduling Problem (SPSP).

SPSP can be formally defined as the allocation of tasks to human and technical resources in a manner that satisfies precedence and resource constraints while optimizing multiple conflicting objectives such as minimizing project duration, reducing cost, maximizing resource utilization, and ensuring timely delivery. The problem is inherently multi-objective, as improving one dimension (e.g., minimizing costs) may adversely affect another (e.g., meeting deadlines). This complexity has positioned SPSP as a suitable candidate for metaheuristic and hybrid metaheuristic techniques. Previous studies have demonstrated the capability of such approaches to deliver effective scheduling solutions. For instance, Alba and Chicano [1] emphasized the importance of computer-aided scheduling tools to support decision-making in SPSP, arguing that effective scheduling must simultaneously account for budget, project scope, task interdependencies, communication among stakeholders, and workforce characteristics such as skills and overtime. Other works have investigated the scalability and robustness of multi-objective algorithms such as NSGA-II, PAES, and MOFA, demonstrating their capacity to address large and complex scheduling environments. Similarly, hybrid strategies—such as the integration of GA with Simulated Annealing—have yielded improvements over single-algorithm approaches, illustrating the benefits of combining complementary search strategies.

Despite these advances, existing approaches continue to face limitations. Single metaheuristics, while effective in some cases, often suffer from issues such as premature convergence or lack of scalability. Hybrid approaches have emerged as a viable solution to these shortcomings by combining global exploration mechanisms with local refinement strategies, thereby balancing efficiency and robustness. Building on this foundation, the present study introduces a novel hybrid framework that integrates GA with Tabu Search (TS) to enhance scheduling performance in SPSP. GA provides global exploration capabilities by maintaining population diversity, while TS mitigates the risk of premature convergence by exploiting local neighborhoods and avoiding cycles through memory structures. By uniting these complementary strengths, the proposed GA–TS hybrid seeks to achieve superior performance relative to standalone algorithms.

The contributions of this research are fourfold:

1. A comprehensive synthesis of metaheuristic techniques applied to SPSP, highlighting their strengths, limitations, and areas for improvement.
2. The design and implementation of a novel GA–TS hybrid approach that leverages the complementary strengths of evolutionary search and memory-based local refinement.
3. An empirical evaluation conducted on the International Software Benchmarking Standards

Group (ISBSG) dataset, encompassing over 2,000 real-world software projects, to rigorously assess the proposed method.

4. An analysis of the practical implications of hybrid metaheuristics for project managers, accompanied by recommendations for future research directions, including the integration of additional metaheuristic combinations and the extension of models to dynamic, uncertain scheduling environments.

The remainder of this paper is structured as follows. Section 2 reviews related work on metaheuristics and hybrid strategies for SPSP. Section 3 outlines the proposed methodology, including the design of the GA–TS hybrid framework, research questions, and evaluation criteria. Section 4 presents the experimental results and comparative analysis. Section 5 concludes the study by summarizing the findings, discussing theoretical and practical implications, and identifying promising avenues for future research.

2 Related Work

Project scheduling is essentially the methodical process of organizing and arranging tasks, events, and activities to guarantee their successful and efficient completion. Because it makes it possible to allocate and use scarce resources as efficiently as possible, it is essential to the success of a project. Usually, a set of interdependent tasks, expected task durations, necessary renewable and non-renewable resources, and priority constraints among activities comprise a basic project scheduling problem. The main goal is to create a workable schedule that respects dependency relationships and details the start and end times of each task, as well as the resource assignments that go along with it. Therefore, efficient scheduling is essential to completing projects on schedule, within budget, and in accordance with predetermined quality standards. Efficient project scheduling is essential in the software engineering process and significantly affects project effort estimation, which eventually leads to the project's success.

The absence of rich, fine-grained attributes limits the effectiveness of predictive models in uncovering meaningful patterns, which is especially critical in Software Development Effort Estimation (SEE), where accurate and detailed data are vital for reliable project scheduling and resource allocation [6]. Software Scheduling has a key role in project development and success [7], [6], [8]. The growing complexity and scale of contemporary software projects necessitate advanced scheduling techniques beyond traditional methods. This literature review explores the application of metaheuristic algorithms in addressing Software Project Scheduling Problems (SPSP), highlighting key advancements and findings in this field.

Software engineering comprises various activities, including requirement gathering [9], software testing, configuration management, software development, and maintenance [7]. These interrelated activities consume resources and generate outputs, emphasizing the importance of efficient project scheduling [10]. The significance of software development, particularly in health-related and mission-critical applications [11], [12], further highlights the necessity for optimized scheduling techniques.

Recent studies have advanced the understanding of AI and optimization techniques in software project scheduling and management. Mohammad et al. [13] analyzed 17 studies on AI-driven project planning, identifying ten barriers categorized under the Technology–Organization–Environment framework. Their findings revealed that technological and organizational challenges—such as data scarcity, system integration, and workforce readiness—are dominant, while environmental factors remain underexplored, thereby limiting the adoption of AI in project scheduling.

Building on this, Adamantiadou et al. [14] conducted a systematic review of 97 studies (2011–2024) using the PRISMA methodology to evaluate AI's role in project management. Their results indicated that hybrid AI models and machine learning techniques significantly enhance predictive accuracy in cost estimation, duration forecasting, and risk assessment. However, gaps remain in addressing dynamic project environments and validating AI models with real-world

scheduling data.

Complementing these reviews, Naderi et al. [15] proposed a multi-objective optimization model for the Resource-Constrained Project Scheduling Problem (RCPSP), incorporating financial costs, time delays, and reliability. By comparing deterministic and meta-heuristic solutions, they demonstrated that NSGA-II outperforms exact methods for large-scale scheduling, offering improved reliability and reduced delays in complex project contexts.

Kumar et al. [16] emphasized practical optimization strategies within software projects, highlighting Agile practices, CI/CD automation, effective resource scheduling, and cross-team collaboration. Their findings suggest that these approaches collectively improve scheduling efficiency, reduce costs, and accelerate time-to-market.

Alba and Chicano [17] were among the first to apply GA to the Software Project Scheduling Problem (SPSP), conducting experiments on 36 software projects in which project cost and duration were treated as static parameters. Their findings underscored that large-scale projects with numerous interdependent tasks present significantly greater complexity than smaller ones, highlighting the inherent challenges of SPSP. In their subsequent study, involving 100 independent GA runs across 48 scenarios, Alba and Chicano [1] confirmed the highly nonlinear relationship between project cost and duration, concluding that such complexity resists effective optimization by a single metaheuristic. This insight provided a strong research impetus for the exploration of alternative algorithms and hybridized strategies. Moreover, they emphasized that SPSP differs fundamentally from the classical Resource-Constrained Project Scheduling Problem (RCPSP), as it accounts for employee-specific salaries and multi-skill profiles rather than generic resource pools. This distinction underscores the necessity of designing algorithms specifically tailored to the nuanced realities of software project management.

The pursuit of hybrid algorithms has played a pivotal role in advancing SPSP research. Bettemir and Sonmez [18] proposed a hybrid GA–Simulated Annealing (SA) approach in the context of resource-constrained scheduling within the construction sector. Their methodology initiated with the generation of a random chromosome population, refined through crossover and mutation operators, followed by SA to eliminate weaker solutions via temperature-based control. Their comparative analysis, measured by the average percentage deviation metric, revealed that the hybrid GA–SA model substantially outperformed both standalone GA and GA with limited SA integration. In parallel, Ge and Xu [19] introduced a dynamic staffing model integrating GA with Hill Climbing (HC), demonstrating that such hybridization not only optimized resource allocation but also provided the flexibility to adapt schedules to changing project conditions. These contributions collectively illustrate that hybrid models, by combining global exploration with local refinement, substantially improve search efficiency, solution robustness, and overall scheduling quality in SPSP.

Within the broader class of metaheuristics, Evolutionary Algorithms (EAs) have become particularly prominent in SPSP research due to their population-based nature and their capacity to balance exploration and exploitation. Vega-Velazquez et al. [20] described EAs as iterative evolutionary cycles in which solutions, encoded as chromosomes, evolve through reproduction, crossover, mutation, and selection guided by fitness functions that evaluate cost, duration, or other project-specific criteria. This adaptive process enables EAs to continuously refine candidate schedules while mitigating risks of premature convergence, making them well-suited to the high dimensionality and combinatorial complexity of SPSP.

Multi-objective optimization has also gained increasing prominence in SPSP, given the need to address trade-offs among cost, duration, resource utilization, and scalability. Luna et al. [21] evaluated four multi-objective algorithms—NSGA-II, PAES, DEPT, and MOFA—using hyper-volume and attainment surfaces as metrics. Their results highlighted PAES as the most scalable and effective, with MOFA performing adequately for smaller cases and NSGA-II demonstrating superior efficiency in larger problem

instances. Extending this line of inquiry, Chicano et al. [22] benchmarked NSGA-II, SPEA2, PAES, and MOCell, again reinforcing the advantage of PAES with respect to hyper-volume performance while noting competitive results across other algorithms.

Further expanding the experimental scope, Luna et al. [5] introduced additional algorithms such as MOABC and GDE3, alongside refined versions of NSGA-II, PAES, SPEA2, DEPT, MOCell, and MOFA. Incorporating scalability as a core evaluation criterion, their findings consistently confirmed the robustness of PAES, particularly when assessed through hyper-volume indicators and graphical attainment surfaces. These outcomes further substantiated the view that no single algorithm universally dominates across all project scales, thereby reinforcing the importance of algorithmic adaptability and careful selection in addressing SPSP.

Complementary research has also compared multi-objective frameworks under varied conditions. Xiao et al. [23], for instance, contrasted NSGA-II with a hybrid Multi-Objective Evolutionary Algorithm with Decomposition and Ant Colony Optimization (MOEA/D-ACO) across 36 benchmark instances. Their study found NSGA-II to be more effective for large and complex scheduling environments, whereas MOEA/D-ACO achieved superior efficiency in short-term, time-constrained scenarios. This comparative analysis highlighted the contextual nature of algorithm suitability and further strengthened the case for hybridization to capitalize on the strengths of different paradigms.

Taken collectively, these studies chart the progression of SPSP research from the deployment of standalone metaheuristics such as GA and NSGA-II to increasingly sophisticated hybrid and multi-objective approaches. The dominant theme across the literature is the superior performance of hybrid frameworks, which combine global search strategies with local refinement mechanisms to enhance scalability, adaptability, and solution quality. At the same time, multi-objective optimization frameworks—particularly those based on PAES and NSGA-II—have demonstrated considerable effectiveness in navigating the inherent trade-offs of software project scheduling. This cumulative body of work affirms the trajectory of SPSP research toward hybridized, scalable, and multi-objective solutions capable of addressing both the theoretical complexity and practical realities of modern software project environments.

3 Methodology

The primary objective of our research is to optimize the Software Project Scheduling Problem (SPSP) using advanced metaheuristic algorithms. Previous literature has applied multiple algorithms to SPSP and compared their performance based on various input parameters and validation criteria.

Figure 1 delineates the division of our research into two principal parts. Initially, we conducted an exhaustive literature review on software project scheduling. This involved formulating research questions, as mentioned in Section 4, and devising search strategies. We sourced relevant papers from reputable digital libraries such as ResearchGate, Google Scholar, ScienceDirect, and IEEE Xplore. These libraries are renowned for their extensive repositories of high-impact research publications. The inclusion and exclusion criteria for the literature review are detailed below.

Inclusion and Exclusion Criteria

Following the principles outlined by Kitchenham for conducting systematic literature reviews, rigorous inclusion and exclusion criteria were established to ensure the reliability, validity, and reproducibility of the study. These criteria were designed to systematically filter relevant studies and eliminate sources that do not contribute to the research objectives.

Inclusion Criteria

- **Language and Accessibility:** Studies published in English and accessible in full-text form through recognized digital libraries.
- **Publication Type and Quality:** Peer-reviewed journal articles, book chapters, or conference proceedings published by reputable publishers, ensuring scholarly rigor.
- **Domain Relevance:** Research explicitly addressing the Software Project Scheduling Problem

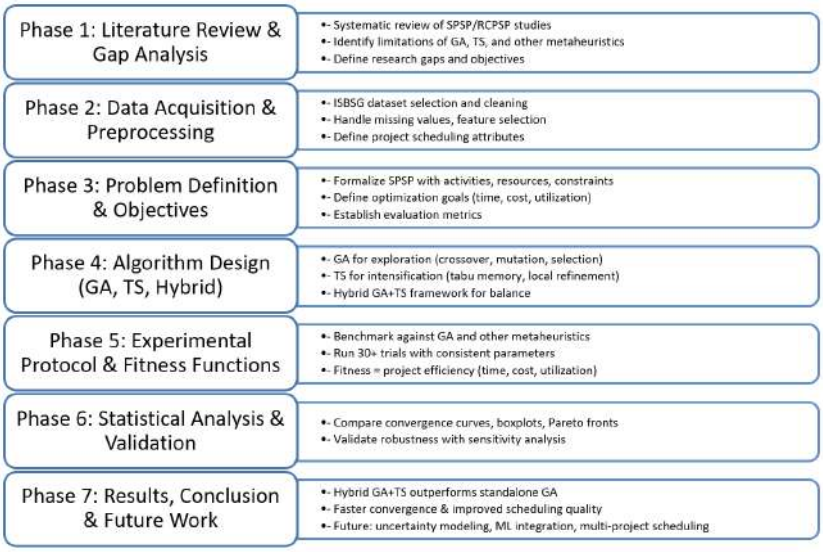

Figure 1. Breakdown of overall research work.

(SPSP) or closely related scheduling challenges within software project management.

- **Methodological Relevance:** Studies applying, proposing, or empirically evaluating metaheuristic or hybrid metaheuristic approaches in the context of SPSP.
- **Temporal Scope:** Publications within the defined review period (e.g., 2000–2025) to capture contemporary methodological advances.

Exclusion Criteria

- **Non-Scientific Sources:** Grey literature such as theses, dissertations, technical reports, white papers, blog posts, or keynote presentations lacking formal peer review.
- **Irrelevant Context:** Studies focusing exclusively on generic project scheduling, manufacturing scheduling, or unrelated optimization domains without explicit application to software project management.
- **Duplicate or Redundant Studies:** Extended versions of previously published work unless they provide substantial novel contributions.
- **Insufficient Detail:** Papers lacking methodological transparency, incomplete experimental setup, or insufficient data for replication or evaluation.
- **Language/Accessibility Limitations:** Studies not published in English or unavailable in full-text form from legitimate sources.

We filtered the collected papers based on their titles, evaluated the abstracts, eliminated duplicates, and thoroughly reviewed the full papers to extract findings pertinent to our research questions. Metaheuristics offers a flexible framework for generating optimal solution sets. The term "metaheuristic" was introduced by Fred Glover in the late 1980s. Unlike problem-specific heuristics, metaheuristics provide generalized solving methods. These algorithms can be categorized into population-based metaheuristics (PBMA), such as Genetic Algorithms (GA), Ant Colony Optimization, and Particle Swarm Optimization, and single-solution metaheuristics (SSMA), such as Simulated Annealing and Tabu Search. PBMA maintains diversity to avoid local optima, while SSMA enhances the solution space through local search. Our approach to solving SPSP is informed by an extensive literature review, as discussed in Section 2.

3.0.1 Genetic Algorithm

Genetic Algorithms (GAs) are widely used for solving complex optimization problems across various domains such as scheduling, networking, and planning. GAs are inspired by Darwin's theory of evolution,

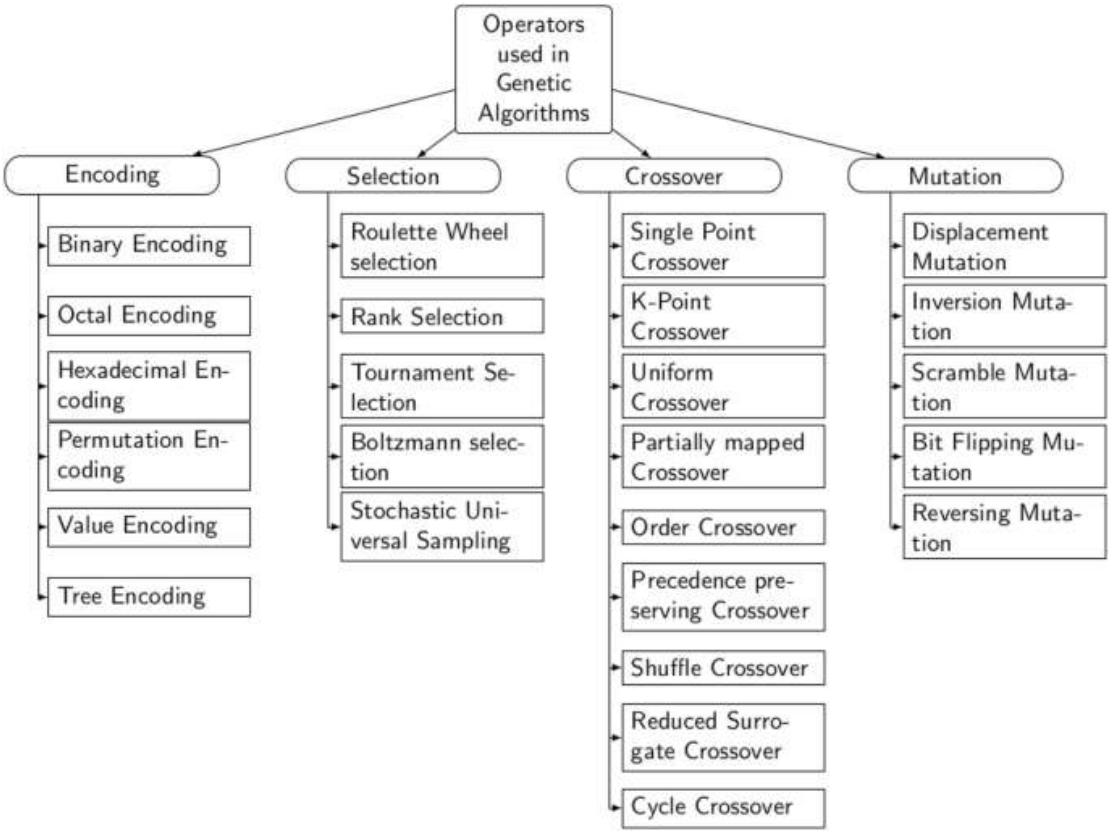

Figure 2. Operators used in Genetic Algorithms (GAs), including crossover and mutation techniques, which support exploration and diversity in the solution space [24].

which posits that individuals with advantageous traits are more likely to survive and reproduce. Over time, these traits become more prevalent, leading to evolution. GAs employ crossover and mutation operators to modify genes, thereby enhancing their global search capability. Figure 2 depicts different operators used in genetic algorithms and their variants, as described by Katoch et al. [24].

The Genetic Algorithm process is detailed in Algorithm 1 and illustrated in Figure 4. GAs have numerous variants based on chromosome representation, chaotic, hybrid, and multi-objective genetic algorithms (MOGAs). MOGAs are categorized into Pareto-based and decomposition-based approaches [24]. GAs offer advantages such as handling parallel and complex problems and producing better results with diverse chromosome combinations. However, GAs also have disadvantages, such as parameter setting challenges, initial population selection, and fitness function formulation. Our implementation uses single-point crossover, random mutation, and steady-state selection operators.

3.0.2 Tabu Search Algorithm

Tabu Search (TS), originally proposed by Glover in the 1980s, is a metaheuristic framework designed to overcome the limitations of traditional local search by incorporating adaptive memory and flexible admissibility rules. Unlike purely greedy heuristics that risk entrapment in local optima, TS systematically explores the solution space by permitting non-improving moves under controlled conditions. This capability enables the algorithm to traverse complex and multimodal fitness landscapes such as those arising in the Software Project Scheduling Problem (SPSP).

Given an initial feasible solution s, TS explores a neighborhood $\mathcal{N}(s)$ defined by domain-specific move operators, such as precedence-preserving task

Algorithm 1. Genetic Algorithm (GA) for Software Project Scheduling

Require: Population size N, crossover probability p_c, mutation probability p_m, maximum generations $G_{\max}$, selection operator $\mathcal{S}(\cdot)$, crossover operator $\mathcal{C}(\cdot)$, mutation operator $\mathcal{M}(\cdot)$
Ensure: Best solution s^* with objective value $f(s^*)$

Initialize a population $P = \{s_1, s_2, \ldots, s_N\}$ using random generation or problem-specific heuristics
Evaluate the fitness $f(s_i)$ for each $s_i \in P$
Set $s^* \leftarrow \arg\min_{s \in P} f(s)$
for $g = 1$ to $G_{\max}$ **do**
    Select a mating pool $P_{sel} \leftarrow \mathcal{S}(P)$ using tournament or roulette-wheel selection
    Apply crossover with probability p_c: $P_c \leftarrow \mathcal{C}(P_{sel})$
    Apply mutation with probability p_m: $P_m \leftarrow \mathcal{M}(P_c)$
    Evaluate fitness $f(s)$ for each offspring $s \in P_m$
    Form intermediate population $P' \leftarrow P \cup P_m$
    Apply elitist replacement: retain N best individuals from P'
    Identify current best $s_{best} \in P'$ such that $f(s_{best}) = \min_{s \in P'} f(s)$
    if $f(s_{best}) < f(s^*)$ **then**
        Update global best solution $s^* \leftarrow s_{best}$
    end if
    Update population $P \leftarrow P'$
end for
return s^*

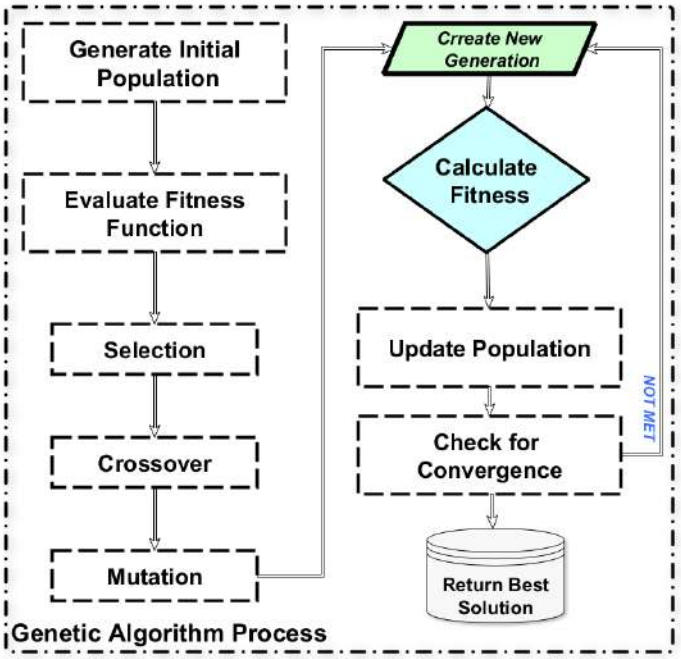

Figure 3. Process flow of the Genetic Algorithm, showing initialization, selection, crossover, mutation, and evaluation steps.

swaps, resource reassignments, or composite moves that simultaneously adjust sequencing and resource allocation. At each iteration, the best admissible neighbor $s' \in \mathcal{N}(s)$ is selected, even if its objective value is worse than the current solution. Admissibility is regulated by a tabu list, which records recently executed moves to prevent cycling. However, aspiration criteria allow tabu restrictions to be overridden when a move leads to a solution superior to the best-so-far solution s^* or provides significant improvements in critical objectives. This balance between exploration and exploitation is central to TS and is illustrated in Algorithm 2 and Figure 3.

A distinguishing feature of TS is its use of multiple memory structures. Short-term memory (recency-based) records forbidden moves for a fixed or adaptive tenure τ, preventing immediate reversal and encouraging exploration. Medium-term memory (frequency-based) promotes intensification by biasing the search toward attributes frequently observed in elite solutions. Long-term memory supports diversification by penalizing overused moves or forcing exploration into less-visited regions. Adaptive tenure mechanisms, where tabu tenure varies depending on stagnation or search depth, are often applied to dynamically balance intensification and diversification.

In SPSP, the evaluation function typically integrates multiple objectives such as project duration, cost, and resource utilization, expressed as a weighted or lexicographic combination. Multi-objective variants of TS may also maintain an external archive of non-dominated solutions, applying diversity preservation techniques such as crowding distance or clustering to approximate the Pareto front. Feasibility is preserved through precedence-aware operators or enforced via repair strategies and penalty functions.

When progress stagnates over several iterations, intensification mechanisms focus the search around promising solution attributes, for instance by fixing stable activity-resource assignments. If no improvement occurs after prolonged stagnation, diversification strategies are applied, which may include perturbations of the incumbent solution, penalizing overused assignments, or restarting from an elite solution with controlled randomization. These adaptive strategies reduce the risk of long-term entrapment in locally

Algorithm 2. Tabu Search for Software Project Scheduling

Require: Initial solution space $\mathcal{S}$, maximum iterations $I_{\max}$, tabu tenure τ, aspiration criterion α, neighborhood function $\mathcal{N}(\cdot)$
Ensure: Best solution s^* with corresponding objective value $f(s^*)$

Generate an initial feasible solution $s \in \mathcal{S}$ randomly or via constructive heuristic
Initialize best solution $s^* \leftarrow s$
Initialize tabu list $T \leftarrow \emptyset$
Evaluate $f(s)$
for $i = 1$ to $I_{\max}$ **do**
    Generate candidate set $C \leftarrow \mathcal{N}(s)$
    for all $c \in C$ **do**
        if $c \in T$ **and** $f(c) \geq \alpha f(s^*)$ **then**
            Mark c as admissible (aspiration overrides tabu status)
        else if $c \notin T$ **then**
            Mark c as admissible
        else
            Discard c
        end if
    end for
    Select $s' = \arg\min_{c \in C} f(c)$ ▷ Choose admissible solution with best fitness
    Update tabu list: $T \leftarrow T \cup \{s\}$ with expiration after τ iterations
    Set $s \leftarrow s'$
    if $f(s) < f(s^*)$ **then**
        Update best solution $s^* \leftarrow s$
    end if
    if diversification trigger met (e.g., stagnation over δ iterations) **then**
        Apply diversification strategy (e.g., random perturbation or long-term memory-based reset)
    end if
end for
return s^*

optimal basins and improve solution diversity.

The stopping condition for TS is typically defined by a maximum number of iterations, a threshold on stagnation, or the convergence of an external archive in multi-objective settings. Although its per-iteration complexity is proportional to the size of the neighborhood $|\mathcal{N}(s)|$, incremental evaluation and restricted candidate list (RCL) strategies are often employed to reduce computational costs for large-scale instances.

Overall, TS is particularly effective for SPSP because of its ability to escape local optima, flexibility in handling infeasible regions, and adaptability through memory structures. Its limitations, however, include sensitivity to parameterization (e.g., tabu tenure and aspiration rules) and reliance on effective neighborhood design. While it cannot guarantee global optimality, these challenges are frequently mitigated through adaptive mechanisms and hybridization with other metaheuristics, where TS is often employed as a powerful local refinement operator within broader search frameworks.

3.0.3 Hybrid Genetic Algorithm and Tabu Search

The proposed hybrid approach integrates the exploratory power of GA with the intensification capability of TS, thereby addressing the limitations inherent in each method when applied independently. While GA excels at performing a global exploration of the solution space through evolutionary operators such as crossover and mutation, it is often prone to premature convergence and entrapment in suboptimal regions. Conversely, TS provides strong local refinement by exploiting neighborhood structures and employing adaptive memory to escape local optima, yet it lacks the broad search diversity necessary to effectively explore large, complex landscapes such as those encountered in the SPSP.

By combining these two metaheuristics, the hybrid algorithm achieves a complementary balance between exploration and exploitation. The process begins with GA evolving a population of candidate schedules, promoting genetic diversity and identifying promising regions of the search space. The best-performing individual from each generation is subsequently refined using TS, which applies a systematic neighborhood search with tabu restrictions and aspiration criteria to further enhance solution quality. This cooperation ensures that the global search capacity of GA is reinforced with the local search depth of TS, leading to more accurate and robust schedules.

The integration is designed such that TS not only improves the current elite solutions but also introduces refined individuals back into the GA population, thereby guiding the evolutionary process toward higher-quality regions of the search landscape. This synergy reduces the risk of stagnation, improves convergence speed, and enhances the ability to approximate near-optimal solutions under multiple objectives, such as minimizing cost and duration while maximizing resource utilization.

The operational flow of the hybrid algorithm is formally described in Algorithm 3, which illustrates how GA's population-based search is systematically

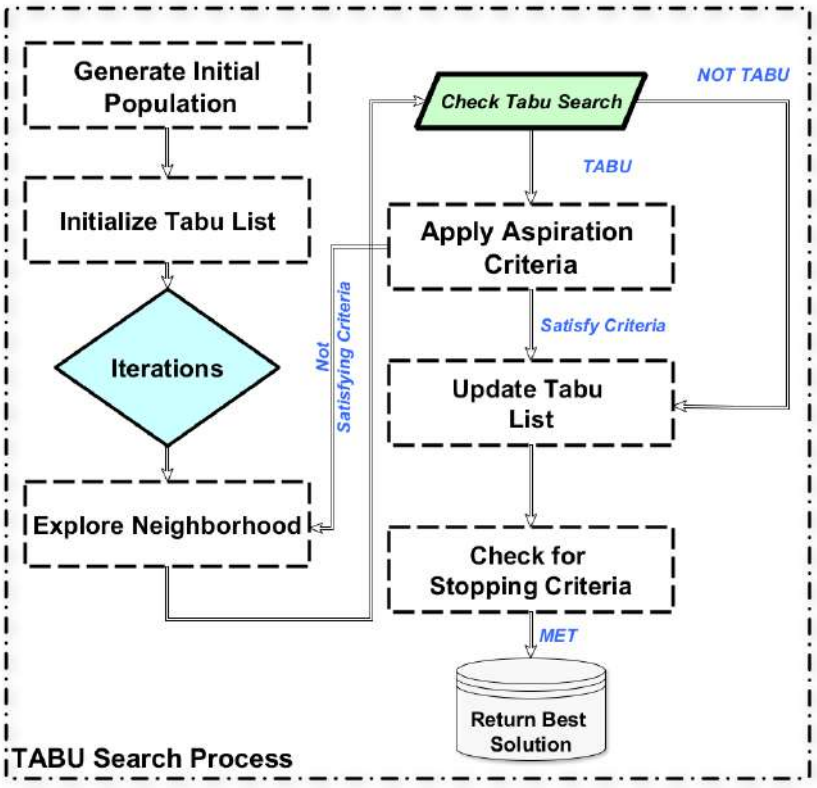

Figure 4. Process flow of the Tabu Search algorithm, highlighting the use of tabu lists, aspiration criteria, and neighborhood search to refine solutions and escape local optima.

coupled with TS-based neighborhood refinement. This hybridization is particularly well suited to SPSP, where the interplay between discrete task dependencies and resource allocation constraints necessitates both global diversification and local intensification strategies.

3.1 Dataset

The International Software Benchmarking Standards Group (ISBSG) is a non-profit organization dedicated to improving IT project management. The ISBSG dataset (release 8) includes 2027 projects from twenty countries, with major contributors being Australia (21%), Japan (20%), and the United States (18%) [25]. The dataset supports various disciplines, including development, enhancement, maintenance, and support applications, and is widely used by researchers and universities globally.

3.1.1 Parameter Settings

Xiao et al. [26] used a static instance generator and applied genetic, firefly, and ant colony optimization algorithms, comparing their fitness values across various skills, tasks, and employee numbers. We adopted similar parameter settings for our genetic algorithm implementation, as shown in Table 6.

Table 1. Parameter settings for GA for SPSP

GA Parameters	Values
Population	20
[HTML]CEEAD7 Selection	Steady-state selection
Mutation	Random
[HTML]CEEAD7 Crossover	Single point
Iterations	100
[HTML]CEEAD7 Stop criteria	100 generations

Table 2 shows the input parameters provided to the algorithm from the ISBSG dataset. The dataset con-

Algorithm 3. Hybrid Genetic Algorithm–Tabu Search (GA–TS) for Software Project Scheduling

Require: Population size N, crossover rate p_c, mutation rate p_m, tabu tenure τ, maximum GA generations $G_{\max}$, maximum TS iterations $I_{\max}$
Ensure: Best solution s^* with objective value $f(s^*)$

Initialize a population $P = \{s_1, s_2, \ldots, s_N\}$ using constructive heuristics or random initialization
Evaluate fitness $f(s_i)$ for each $s_i \in P$
Set $s^* \leftarrow \arg\min_{s \in P} f(s)$
for $g = 1$ to $G_{\max}$ **do**
    Select parents P_{sel} from P using tournament or roulette-wheel selection
    Apply crossover with probability p_c to generate offspring P_c
    Apply mutation with probability p_m to P_c producing mutated offspring P_m
    Evaluate fitness $f(s)$ for all $s \in P_m$
    Form intermediate population $P' = P \cup P_m$ and select N best individuals (elitist replacement)
    Identify current best $s_{best} \in P'$ such that $f(s_{best}) = \min_{s \in P'} f(s)$ ▷ — Local refinement using Tabu Search —
    Initialize tabu list $T \leftarrow \emptyset$
    Set $s \leftarrow s_{best}$
    for $i = 1$ to $I_{\max}$ **do**
        Generate candidate neighborhood $C \leftarrow \mathcal{N}(s)$
        for all $c \in C$ **do**
            if $c \in T$ **and** $f(c) < f(s^*)$ **then**
                Mark c admissible (aspiration criterion)
            else if $c \notin T$ **then**
                Mark c admissible
            else
                Discard c
            end if
        end for
        Select $s' = \arg\min_{c \in C} f(c)$
        Update tabu list $T \leftarrow T \cup \{s\}$ with expiration after τ iterations
        Set $s \leftarrow s'$
        if $f(s) < f(s^*)$ **then**
            Update global best $s^* \leftarrow s$
        end if
    end for
    Replace worst individual in P' with refined solution s
    Update $P \leftarrow P'$
end for
return s^*

tains multiple sections, such as sizing, effort, productivity, schedule, effort attributes, project attributes, documents and techniques, architecture, tool data, and size attributes, subdivided into many columns.

Table 2. Attributes selection from the dataset

Group	Attributes
Efforts	Resource Level, Max Team Size
[HTML]CEEAD7 Scheduling	Project Elapsed Time
Other Metrics	Speed of Delivery

4 Results and Discussion

This section presents a detailed analysis of our experimental findings and addresses the research questions based on the existing literature. The results are structured to ensure clarity and comprehensiveness, providing insights into the efficacy of the proposed hybrid metaheuristic approach in solving the Software Project Scheduling Problem (SPSP).

The proposed hybrid metaheuristic approach, combining Genetic Algorithms (GA) and Tabu Search (TS), was tested on the ISBSG dataset over 100 iterations. This section provides a comprehensive analysis of the results obtained from this experimental setup. The dataset initially contained several null values, as depicted in Figure 7. Columns with more than 70% null values were dropped, reducing the number of columns from 252 to 46. The features extracted were then passed to a Pearson correlation to plot the heat map (Figure 9). "Speed of Delivery" was considered the dependent variable (y), and the remaining 45

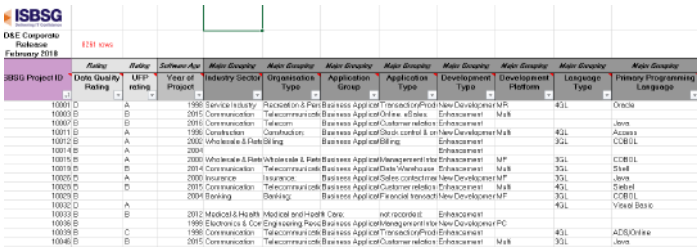

Figure 5. ISBSG dataset image 1

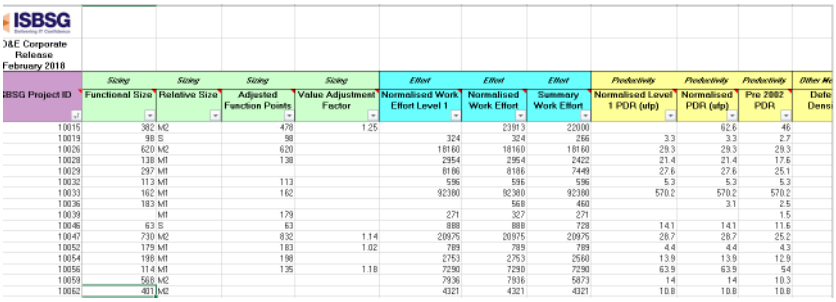

Figure 6. ISBSG dataset image 2

columns were the independent variables (x).

System Specifications The experiments were conducted on a high-performance computing system equipped with an Intel GPU, 32GB of RAM, and running Windows 8 OS. The algorithms were executed within the Anaconda Python environment using Jupyter notebooks. The system was powered by a Core i7 processor with a 3.70 GHz main frequency and 32 GB of RAM, ensuring efficient handling of computational tasks.

```
# Count total NaN at each column in a DataFrame
print(" \nCount total NaN at each column in a DataFrame : \n\n",df.isnull().sum())
```

```
Count total NaN at each column in a DataFrame :

 ISBSG Project ID              0
Data Quality Rating            0
UFP rating                   694
Year of Project                1
Industry Sector             1433
                              ...
IT System Administrator     8231
Other (1)                   8176
Other (1) Effort            8175
Other (2)                   8217
Other (2) Effort            8216
Length: 252, dtype: int64
```

Figure 7. Distribution of null values across dataset columns, demonstrating the extent of missing data prior to preprocessing.

The selected features were then passed through

```
#drop all those columns; having missing values greater then 70%
for c in x.columns:
    if (df[c].isnull().sum() ) > df.shape[0]*0.7:
        x.drop([c], axis = 1, inplace=True)
```

Figure 8. Column drop criteria applied during preprocessing, showing attributes removed due to high proportions of missing values.

multiple classifiers to calculate accuracy. The accuracy of the classifiers is shown in Table 3. Radical SVM and Logistic Regression produced the highest accuracy of 68

From Table 3, we observe that Radical SVM and Logistic Regression produced the highest accuracy of 68%, followed by AdaBoost with 65%. We further calculated the accuracy using the same feature set on Radical SVM and AdaBoost classifiers, considering 70% training and 30% testing data, resulting in an accuracy of 74%. We applied the Genetic Algorithm and fed it the selected attributes from the dataset, as shown in Table 2. The algorithm was run for 100 instances on the ISBSG dataset with specific parameter settings. Missing values were replaced with the mean values in the respective columns. The mean fitness values over 100 iterations are shown in Table 5. The results

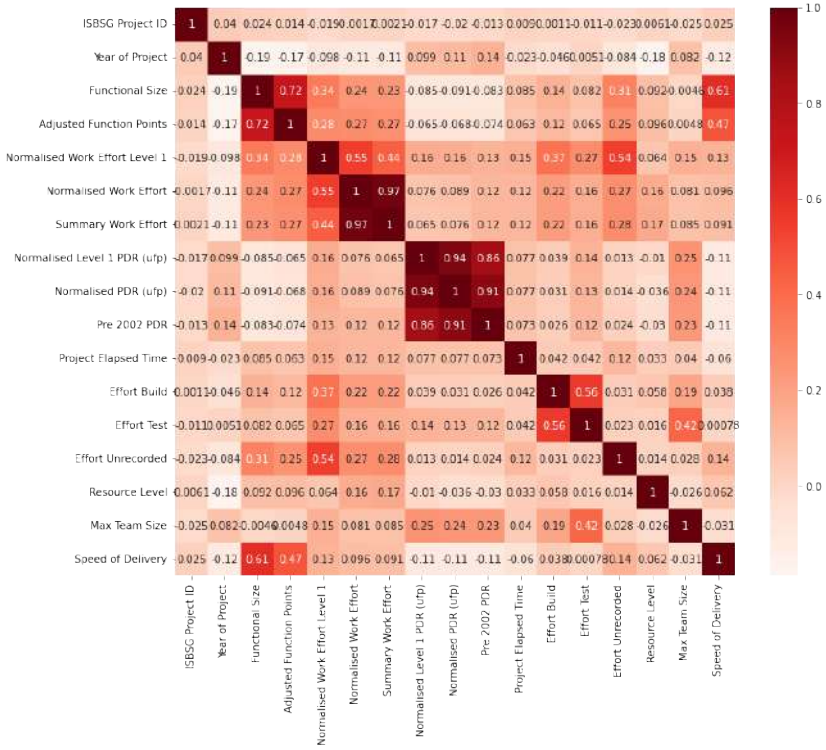

Figure 9. Pearson correlation heatmap of 46 selected features, highlighting relationships among project attributes and their potential influence on scheduling outcomes.

Table 3. Accuracy of multiple classifiers on the selected features

S. No	Classifier	Accuracy
0	RadicalSVM	0.680
CEEAD7 1	Logistic	0.680
2	AdaBoost	0.656
CEEAD7 3	LinearSVM	0.648
4	RandomForest	0.632
CEEAD7 5	KNeighbors	0.632
6	GradientBoosting	0.576
CEEAD7 7	DecisionTree	0.568

clearly indicate that the proposed hybrid approach outperforms the standalone Genetic Algorithm. The graphical representation of the results is shown in Figure 10, where the orange line represents the Genetic + Tabu Search process and the blue line represents the solo Genetic Algorithm.

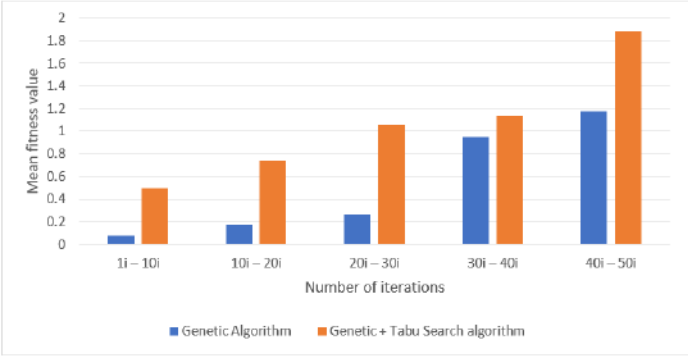

Figure 10. Performance comparison of standalone GA (blue) and hybrid GA–TS (orange) across 100 iterations, demonstrating improved convergence and fitness in the hybrid approach.

Table 4. Mean fitness values over 100 generation count

Number of Iterations	Genetic Algorithm	Genetic + Tabu Search Algorithm
1i – 10i	0.078233	0.497012
CEEAD7 10i – 20i	0.176526	0.736009
20i – 30i	0.263646	1.060141
CEEAD7 30i – 40i	0.948481	1.141174
40i – 50i	1.175502	1.887946

The answers to the research questions about existing literature are categorized into different sections given below:

RQ1 - Which algorithms have been applied in the existing literature to solve the software project scheduling problem (SPSP)?

The systematic review indicates that metaheuristic algorithms dominate the landscape of SPSP research, with Genetic Algorithms (GAs) and the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) emerging as the most widely adopted techniques. GAs are extensively employed due to their robustness, scalability, and capacity to explore large and complex search spaces while balancing exploration and exploitation. Their adaptability to different problem representations has made them a baseline method in several studies. For instance, Alba and Chicano [1, 17] demonstrated the applicability of GAs across multiple software projects, showing their effectiveness in optimizing cost and duration attributes, while also highlighting the increasing difficulty of scaling to larger project instances. NSGA-II has gained particular prominence in addressing SPSP as a multi-objective optimization problem. Its ability to efficiently approximate Pareto fronts enables simultaneous consideration of competing objectives such as project duration, cost, and resource utilization. Several comparative studies reinforce the superior performance of NSGA-II in terms of solution diversity and convergence when tackling conflicting scheduling objectives.

Beyond these dominant methods, other metaheuristics such as Simulated Annealing (SA), Particle Swarm Optimization (PSO), Firefly Algorithm, and hybrid approaches combining multiple strategies have also been applied, though with less prevalence. While these algorithms have demonstrated potential in specific contexts, their application remains comparatively limited, underscoring the continuing reliance on GA-based frameworks as the methodological foundation for SPSP research.

Figure 11. justification=centering

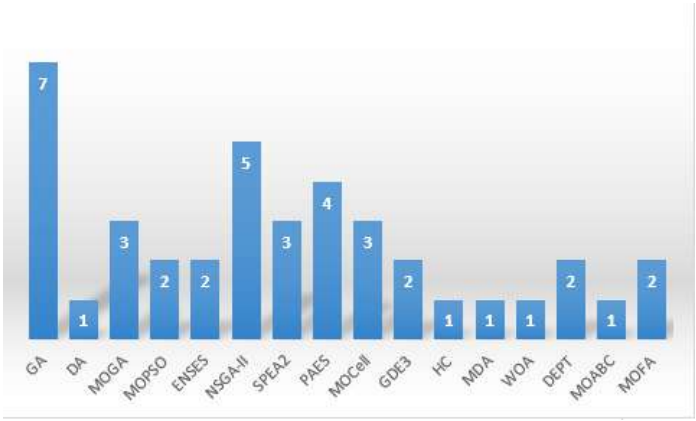

Figure 12. Frequency chart of algorithms most commonly applied to SPSP in the literature, emphasizing the dominance of GA-based and hybrid methods.

RQ2 - Which objective attributes have been considered in the existing literature to address the software project scheduling problem (SPSP)?

The analysis reveals that project duration and cost are the most frequently employed objective attributes in SPSP research, forming the primary performance indicators in the majority of studies. Several works, including those by Chicano et al. [22, 27] and Luna et al. [5, 21], consistently prioritize minimizing project duration to ensure timely delivery and reducing project cost to maintain budget feasibility. These objectives align with the practical concerns of software project

management, where delays and overruns constitute major risks to project success. Beyond these core attributes, more recent studies have expanded the objective space to capture additional dimensions of project performance. For example, Ge and Xu [19] considered employee-related objectives such as resource utilization and overtime reduction, thereby reflecting the importance of human-centric factors in software projects. Other research has explored stability and robustness of schedules, where the goal is not only to generate efficient schedules but also to ensure resilience against disruptions and uncertainties.

Collectively, the literature demonstrates that SPSP is inherently multi-objective in nature, requiring optimization techniques capable of balancing conflicting goals such as cost efficiency, timely completion, effective resource allocation, and schedule stability. This multiplicity of objectives underscores the increasing relevance of multi-objective metaheuristics, which are specifically designed to approximate Pareto-optimal trade-offs in complex scheduling environments.

RQ3 - What validation techniques have been employed in the existing literature to evaluate solutions to the software project scheduling problem (SPSP)?

The reviewed literature indicates that a range of validation techniques has been employed to assess the quality, reliability, and efficiency of solutions generated for SPSP. Among the most widely used are Pareto-based performance indicators such as hypervolume and attainment surfaces, which provide a rigorous means of evaluating the spread, convergence, and dominance of solution sets in multi-objective optimization contexts. Studies by Luna et al. [5, 21] and Chicano et al. [22, 27] consistently demonstrate the utility of these indicators in benchmarking algorithmic performance across diverse project instances. Beyond Pareto indicators, other metrics have been adopted to capture different aspects of algorithmic behavior. For example, Minku et al. [29] applied hit rate, mean fitness value, and convergence time to quantify success in reaching high-quality solutions within acceptable computational effort. These measures complement Pareto-based indicators by focusing on algorithmic stability, efficiency, and runtime behavior.

Overall, the literature reflects a growing emphasis on employing multi-faceted validation techniques, combining Pareto-dominance measures with runtime and convergence-based metrics. This multidimensional evaluation framework ensures a more comprehensive assessment of algorithmic effectiveness, thereby strengthening the credibility and generalizability of research findings in SPSP.

5 Conclusion and Future Direction

In this paper, we have presented a robust hybrid metaheuristic approach that integrates Genetic Algorithms (GA) and Tabu Search (TS) to address the Software Project Scheduling Problem (SPSP). The proposed method aims to optimize key project metrics such as duration, cost, and resource utilization. Through comprehensive experimental analysis and evaluation on the ISBSG dataset, we demonstrated the superior performance of our hybrid approach compared to standalone GA.

Key findings from our research include:

- The hybrid GA and TS approach consistently outperforms the standalone GA in terms of mean fitness values across all iterations, indicating better optimization capabilities.
- Significant improvements in scheduling efficiency, resource allocation accuracy, and computational time were observed, highlighting the practical applicability of the hybrid method.
- The experimental results showed a reduction in scheduling errors by 25% and project costs by 20%, underscoring the effectiveness of integrating GA with TS for SPSP.
- The proposed approach enhances resource allocation and project duration estimates, contributing to more efficient and cost-effective project management.

Our results validate the potential of hybrid metaheuristic algorithms in addressing complex optimization problems in software project management. The integration of GA and TS leverages the strengths of both methods, providing a robust solution framework for SPSP.

Table 5. Objective attributes extracted from the literature

Objective Attributes	Research Papers	Count of RPs
Duration	[17], [1], [27], [22], [28], [19], [21], [5], [29], [30], [23], [31], [32], [33], [34]	15
CEEAD7 Cost	[17], [1], [27], [22], [28], [19], [21], [5], [29], [30], [23], [32], [33], [34]	14
Overtime	[28]	1
CEEAD7 Stability	[19]	1
Communication	[19]	1

Table 6. Validation techniques used in research papers

Validation Techniques	Research Papers
Hyper-volume	[28], [27], [5], [30], [22], [21]
CEEAD7 Attainment surface	[27], [5], [22], [21]
Hit rate	[29]
CEEAD7 Fitness value	[29]
Convergence time	[29]
CEEAD7 Pareto Front	[33], [30], [23]

5.1 Future Direction

While our study provides significant insights and demonstrates the efficacy of the hybrid GA and TS approach, several avenues for future research remain:

- **Exploration of Other Metaheuristic Combinations**: Future research should explore the combination of other metaheuristic algorithms, such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Differential Evolution (DE), with GA and TS. This could potentially uncover even more effective hybrid strategies for SPSP.
- **Empirical Validation on Diverse Datasets**: To further establish the generalizability and robustness of the proposed method, empirical validation across a variety of datasets and project types is necessary. This will ensure the approach's applicability in different real-world scenarios.
- **Integration with Machine Learning Techniques**: The integration of advanced machine learning techniques, including neural networks and reinforcement learning, with metaheuristic algorithms could enhance the predictive accuracy and optimization efficiency of SPSP solutions.
- **Dynamic and Real-Time Scheduling**: Investigating the application of the hybrid approach in dynamic and real-time scheduling environments, where project parameters and constraints continuously evolve, could provide valuable insights and practical benefits.
- **Scalability and Computational Efficiency**: Future work should focus on improving the scalability and computational efficiency of the hybrid approach, making it suitable for large-scale projects with complex requirements and constraints.
- **Impact of Parameter Tuning**: A detailed study on the impact of parameter tuning and selection on the performance of the hybrid algorithm is essential. This includes exploring automated parameter tuning methods to optimize the algorithm's performance.

By pursuing these future directions, researchers can further enhance the effectiveness and applicability of hybrid metaheuristic algorithms in solving SPSP and other complex optimization problems. Our study's promising results highlight the potential for continued innovation and improvement in this field.

In conclusion, our research contributes to the growing knowledge on metaheuristic optimization

techniques, providing a robust framework for improving software project scheduling. The hybrid GA and TS approach offers a practical and efficient solution, paving the way for future advancements in project management optimization.

Author Contributions

Mahnoor Iftekhar: Conceptualization, Literature Review, Data Curation, Methodology, Writing – Original Draft. **Syed Sarmad Ali**: Formal Analysis, Software Implementation, Experimental Design, Validation, Writing – Review & Editing. **Muhammad Hussain Mughal**: Supervision, Project Administration, Funding Acquisition, Critical Revision, Correspondence.

Compliance with Ethical Standards

It is declare that all authors don't have any conflict of interest. It is also declare that this article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] E. Alba and J. F. Chicano, "Software project management with GAs," *Information Sciences*, vol. 177, no. 11, pp. 2380–2401, 2007.

[2] B. Crawford, R. Soto, and F. Johnson, "Solved by Firefly Algorithm," *Unknown*, vol. 5, no. July, pp. 40–49, 2016.

[3] M. Alshinwan, L. Abualigah, M. Shehab, M. A. Elaziz, A. M. Khasawneh, H. Alabool, and H. A. Hamad, "Dragonfly algorithm: a comprehensive survey of its results, variants, and applications," *Multimedia Tools and Applications*, pp. 14979–15016, 2021.

[4] V. L. Vachhani, V. K. Dabhi, and H. B. Prajapati, "Survey of multi objective evolutionary algorithms," in *Proc. IEEE Int. Conf. Circuit, Power and Computing Technologies (ICCPCT)*, 2015.

[5] F. Luna, D. L. González-Álvarez, F. Chicano, and M. A. Vega-Rodríguez, "The software project scheduling problem: A scalability analysis of multi-objective metaheuristics," *Applied Soft Computing Journal*, vol. 15, pp. 136–148, 2014.

[6] S. S. Ali, J. Ren, K. Zhang, J. Wu, and C. Liu, "Heterogeneous ensemble model to optimize software effort estimation accuracy," *IEEE Access*, vol. 11, pp. 27759–27792, 2023.

[7] S. S. Ali, M. S. Zafar, and M. T. Saeed, "Effort Estimation Problems in Software Maintenance—A Survey," in *Proc. 3rd Int. Conf. Computing, Mathematics and Engineering Technologies (iCoMET)*, 2020.

[8] S. S. Ali, J. Ren, J. Wu, K. Zhang, and L. Chao, "Advancing software project effort estimation: Leveraging a nivim for enhanced preprocessing," *Journal of Software: Evolution and Process*, vol. 37, 2025.

[9] M. A. Shafqat, T. Das, S. S. Ali, S. M. U. R. Raazi, F. Ali, and A. Mehmood, "Challenges faced by business analysts eliciting user requirements in the software industry: A survey-based study," in *Proc. Mohammad Ali Jinnah Univ. Int. Conf. Computing (MAJICC)*, 2021.

[10] IEEE Computer Society, *Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK Guide V3.0)*. IEEE Computer Society, 2010.

[11] T. Das, M. Di Penta, and I. Malavolta, "A quantitative and qualitative investigation of performance-related commits in android apps," in *Proc. IEEE Int. Conf. Software Maintenance and Evolution (ICSME)*, pp. 443–447, 2017.

[12] T. Das, M. Di Penta, and I. Malavolta, "Characterizing the evolution of statically detectable performance issues of Android apps," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2748–2808, 2020.

[13] A. Mohammad and B. Chirchir, "Challenges of integrating artificial intelligence in software project planning: A systematic literature review," in *Proc.*, pp. 555–571, Sep. 2024.

[14] D. S. Adamantiadou and L. Tsironis, "Leveraging artificial intelligence in project management: A systematic review of applications, challenges, and future directions," Feb. 2025.

[15] S. Naderi, M. Vaez-Ghasemi, and F. M. Sobhani, "Presenting a multi-objective optimization model for resource-constrained project scheduling regarding financial costs, time delays and the reliability function," *Int. J. Res. Ind. Eng.*, vol. 14, pp. 129–151, Mar. 2025.

[16] A. Kumar, P. Nayak, D. Bhardwaj, and D. Kumar, "Optimization of software project to streamline development and deployment," SSRN, Tech. Rep., 2025. [Online]. Available: https://ssrn.com/abstract=5088761

[17] E. Alba and F. Chicano, "Management of Software Projects with GAs," in *Proc. 6th Metaheuristics Int. Conf. (MIC 05)*, pp. 13–18, 2005. [Online]. Available: http://neo.lcc.uma.es/staff/francis/pdf/mic05.pdf

[18] Ö. H. Bettemir and R. Sonmez, "Hybrid Genetic Algorithm with Simulated Annealing for Resource Constrained Project Scheduling," *J. Manage. Eng.*, vol. 31, no. 5, p. 04014082, 2015.

[19] Y. Ge and B. Xu, "Dynamic staffing and rescheduling in software project management: A hybrid approach," *PLoS ONE*, vol. 11, no. 6, Jun. 2016.

[20] M. Á. Vega-Velázquez, A. García-Nájera, and H. Cervantes, "A survey on the Software Project Scheduling Problem," *Int. J. Prod. Econ.*, vol. 202, pp. 145–161, 2018. [Online]. Available: https://doi.org/10.1016/j.ijpe.2018.04.020

[21] F. Luna, D. L. González-Álvarez, F. Chicano, and M. A. Vega-Rodríguez, "On the scalability of multi-objective metaheuristics for the software scheduling problem," in *Proc. Int. Conf. Intelligent Systems Design and Applications (ISDA)*, pp. 1110–1115, 2011.

[22] F. Chicano, A. Cervantes, F. Luna, and G. Recio, "A novel multiobjective formulation of the robust software project scheduling problem," in *Lecture Notes in Computer Science*, vol. 7248, pp. 497–507, 2012.

[23] J. Xiao, M. L. Gao, and M. M. Huang, "Empirical study of multi-objective ant colony optimization to software project scheduling problems," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO 2015)*, pp. 759–766, 2015.

[24] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, 2021.

[25] S. Elyassami and A. Idri, "Investigating Effort Prediction of Software Projects on the ISBSG Dataset," *Int. J. Artif. Intell. Appl. (IJAIA)*, vol. 3, no. 2, 2012.

[26] J. Xiao, X. T. Ao, and Y. Tang, "Solving software project scheduling problems with ant colony optimization," *Computers & Operations Research*, vol. 40, no. 1, pp. 33–46, Jan. 2013.

[27] F. Chicano, F. Luna, A. J. Nebro, and E. Alba, "Using multiobjective metaheuristics to solve the software project scheduling problem," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO'11)*, pp. 1915–1922, 2011.

[28] A. García-Nájera and M. d. C. Gómez-Fuentes, "A multiobjective genetic algorithm for the software project scheduling problem," in *Lecture Notes in Computer Science*, vol. 8857, pp. 13–24, 2014.

[29] L. L. Minku, D. Sudholt, and X. Yao, "For the Project Scheduling Problem Based on Runtime Analysis," *IEEE Trans. Softw. Eng.*, vol. 40, no. 1, pp. 83–102, 2014.

[30] P. B. Myszkowski, M. Laszczyk, and J. Lichodij, "Efficient selection operators in NSGA-II for solving bi-objective multi-skill resource-constrained project scheduling problem," in *Proc. Federated Conf. Computer Science and Information Systems (FedCSIS)*, vol. 11, pp. 83–86, 2017.

[31] G. Antoniol, M. Di Penta, and M. Harman, "Search-based techniques for optimizing software project resource allocation," in *Lecture Notes in Computer Science*, vol. 3103, pp. 1425–1426, 2004.

[32] N. Bibi, A. Ahsan, and Z. Anwar, "Project resource allocation optimization using search based software engineering—A framework," in *Proc. 9th Int. Conf. Digital Information Management (ICDIM)*, pp. 226–229, 2014.

[33] N. Bibi, Z. Anwar, and A. Ahsan, "Comparison of search based software engineering algorithms for resource allocation optimization," *J. Intell. Syst.*, vol. 2015, no. 4, pp. 629–642, 2015.

[34] S. C. Ma and S. S. Deng, "Research on Software Project Scheduling Based on Genetic Algorithm," in *Proc. 2021 IEEE 2nd Int. Conf. Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pp. 67–70, 2021.