Contents lists available at ScienceDirect

Smart Health

journal homepage – www.elsevier.com/locate/smhl

A Metaheuristic based Approach for Maximizing Sprint Capacity Utilization using Genetic Algorithm

Md. Nazmul Abdal (0424052094) [a], Ryhan Ahmed Tamim (0424052095) [a], Nayan Chandra Das (0424052071) [a]

[a] *Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh*

ARTICLE INFO

Keywords:
Genetic algorithm, sprint capacity utilization, selection, crossover, mutation

ABSTRACT

Efficient sprint capacity utilization is a crucial factor in optimizing project execution in agile environments. This paper presents a metaheuristic-based approach to maximize sprint capacity utilization using a Genetic Algorithm (GA). The problem is formulated as a task allocation challenge, where 200 tasks, distributed across seven projects, must be assigned to five team members. Each task has a predefined completion time, and each team member has a specific daily capacity and is trained only for a subset of the projects. The objective is to maximize the utilization of each team member within a 14-day sprint, ensuring that their daily capacity limits are respected and tasks are allocated based on their expertise. Our proposed GA-based solution evolves a population of task assignments over generations, optimizing for maximum utilization while adhering to the constraints. This method encodes task assignments as chromosomes and employs selection, crossover, and mutation operators to iteratively refine feasible solutions. The effectiveness of the approach is evaluated through experimental results, demonstrating its capability to enhance sprint capacity utilization while adhering to project-specific constraints. This research provides a scalable and adaptable solution for task allocation in agile project management, offering a balance between computational efficiency and optimal resource utilization.

1. Introduction

In the realm of software development and project management, efficient resource allocation and task scheduling are critical to maximizing productivity and ensuring timely project delivery. Agile methodologies, particularly Scrum, have gained widespread adoption due to their iterative and incremental approach, which emphasizes adaptability and continuous improvement. A key component of Scrum is the sprint, a time-boxed period during which a set of tasks is completed. However, one of the persistent challenges in sprint planning is the optimal utilization of team members' capacities while adhering to constraints such as skill sets, task dependencies, and time limitations [1]. This challenge becomes even more pronounced when dealing with multiple projects and a diverse set of tasks, each with varying complexities and completion times. Resource allocation in Agile project management presents several challenges. First, team members have varying expertise and are trained only in specific projects, limiting the flexibility of task assignments. Second, each member has a predefined working capacity per day, which must be fully utilized without exceeding constraints [2]. Third, with a large number of tasks distributed across multiple projects, finding an optimal assignment that maximizes capacity utilization while adhering to skill constraints is computationally difficult.

The motivation behind this research stems from the need to address the inefficiencies in sprint planning, particularly in scenarios where team members have specialized skills and limited availability. Traditional scheduling methods often fall short in handling the complexity and dynamic nature of such environments. These methods may struggle with complex constraints, making metaheuristic approaches a viable alternative [3]. The ability of GA to explore large search spaces and converge to near-optimal solutions make it particularly well-suited for this problem. GA, inspired by the principles of natural selection and evolution, is particularly well-suited for combinatorial optimization problems, making it an ideal choice for maximizing sprint capacity utilization.

In this paper, we propose a metaheuristic-based approach to maximize sprint capacity utilization using a Genetic Algorithm. The problem is defined as follows: Given a set of 7 projects and 200 tasks, each associated with a specific project and having a defined task hour, the objective is to assign these tasks to 5 team members over a 14-day sprint period. Each team member has a maximum daily capacity in hours and is trained on a subset of the projects. The primary goal is to maximize the utilization of each team member's capacity while ensuring that tasks are only assigned to members trained in the relevant projects. The constraints include the total sprint duration, individual capacity limits, and the skill compatibility between team members and tasks. Formally, let $P = \{p_1, p_2, \ldots, p_7\}$ represent the set of projects, and $T = \{t_1, t_2, \ldots, t_{200}\}$ denote the set of tasks, where each task t_i is associated with a project p_j and has a completion time h_i . Let $M = \{m_1, m_2, \ldots, m_5\}$ be the set of team members, each with a daily capacity c_k and a set of trained projects $S_k \subseteq P$. The objective is to maximize the total utilization of team members' capacities over the sprint period, subject to the constraints that each task is assigned to a member trained in the corresponding project and that no member exceeds their daily capacity. By leveraging the power of GA, this research aims to provide a robust and scalable solution to the sprint capacity utilization problem, ultimately contributing to more efficient and effective project management practices in agile environments. The proposed approach not only addresses the immediate challenge of task allocation but also offers a framework that can be adapted to various other resource optimization problems in software development and beyond.

1.1. Literature Review

The authors in [1] highlighted the importance of effective sprint planning to address the dynamic demands of the market. They developed a multi-objective mixed-integer programming model that optimized Scrum planning by considering three key objectives: maximizing sprint capacity utilization, prioritizing high-priority user stories in primary sprints, and grouping affine user stories within the same sprint. This approach not only advanced the theoretical understanding of Scrum planning but also provided practical insights by incorporating alternative user stories, a feature often overlooked in existing literature. To tackle the computational complexity of large-scale problems, heuristic methods such as the Non-dominated Sorting Genetic Algorithm (NSGA-II) and the Strong Pareto Evolutionary Algorithm (SPEA2) were employed, as traditional optimization techniques proved inadequate for big-sized instances. Performance evaluations using metrics like Hypervolume (HV), Epsilon (ϵ), Generational Distance (GD), Inverted Generational Distance (IGD), Inverted Generational Distance Plus (IGD+), and Spread (Δ) revealed that NSGA-II outperformed SPEA2 in terms of the ϵ indicator for large instances, while SPEA2 demonstrated superior performance in HV, GD, IGD, and IGD+ metrics. However, the results for HV, ϵ, IGD, and IGD+ were closely aligned, suggesting that both algorithms were viable for solving multi-objective Scrum planning problems. The sprint planning phase in agile software development is a critical determinant of project success, as it directly influences the efficiency and effectiveness of task execution. Recent researchers have emphasized the importance of considering multiple factors during sprint planning, such as the estimated complexity, risk values, and business value of user stories, to ensure a balanced and optimal sprint plan. The authors in [4] addressed this challenge by transforming the sprint planning problem into a generalized optimization problem, which was then solved using the IBM ILOG CPLEX optimizer, a powerful tool for linear programming. While CPLEX was capable of providing feasible solutions, the computational time required for large-scale problems often rendered it impractical for operational use. To mitigate this limitation, the authors proposed integrating a Greedy Heuristic approach with CPLEX, demonstrating that the heuristic significantly reduced computation time while maintaining solution effectiveness. Furthermore, this study highlighted the importance of considering multiple dimensions of user stories, such as complexity, risk, and business value, during sprint planning. One notable approach to addressing the sprint planning problem is the use of Integer Linear Programming (ILP) models. The authors of [5] proposed an ILP model that aimed to maximize the business value perceived by users while adhering to development constraints. The model took into account the estimates provided by the project team and sought to find an optimal sprint plan. However, the authors acknowledged that solving the ILP model to optimality using general-purpose Mixed Integer Programming (MIP) solvers, such as IBM Ilog Cplex, could be computationally intensive. For some instances, finding even a feasible solution within a reasonable time frame was impractical for operational use. To overcome these computational challenges, they introduced an effective Lagrangian heuristic. This heuristic was based on a relaxation of the proposed ILP model and incorporated greedy and exchange algorithms to improve solution quality. The Lagrangian heuristic aimed to provide near-optimal solutions in a more computationally efficient manner, making it suitable for real-world applications. The effectiveness of this approach was demonstrated through computational experiments on both real and synthetic projects, highlighting its potential to enhance sprint planning processes. In [6], the authors developed a business process model for information technology-based project implementation using Scrum and proposed an algorithm for Sprint Backlog planning that accounted for uncertainty. Their study formalized tasks related to estimating the labor intensity of user stories and evaluating risks during planning, culminating in a technology for selecting user stories for the Sprint Backlog. Numerical experiments validated the proposed decision-support technology, demonstrating its practical utility in improving decision-making efficiency during sprint planning. The research also introduced a method for evaluating the adequacy of the technology and selected key performance indicators for assessing team performance. Their results confirmed the significance of the proposed technology in mitigating uncertainty, reducing decision-making time, and enhancing the overall sprint planning process. The study concluded by recommending the adoption of this technology in practice, emphasizing its scientific novelty in improving sprint planning under uncertain conditions. Kula et al. [7] investigated how prioritization criteria, such as urgency, sprint goal alignment, and business value, were influenced by project-specific factors like resource availability and client type, as identified through a survey conducted at ING. Recognizing the need for contextual support in sprint planning, the authors proposed an optimization model that generated tailored sprint plans by integrating team-specific objectives and historical sprint data. By applying to real-world data from 4,841 sprints at ING, the model demonstrated significant improvements in team

alignment, sprint plan effectiveness, and overall performance by delivering higher business value, better alignment with sprint goals, and enhanced risk mitigation. Their findings underscored the potential of context-aware optimization methods to enhance the efficiency and outcomes of sprint planning practices, offering a valuable contribution to agile project management. The researchers of [8] proposed a context-aware task allocation decision support system designed to balance quality and timeliness, thereby enhancing the overall utility of Agile software development projects. By formulating the Agile process as a distributed constraint optimization problem, the proposed framework leveraged data collected from Scrum-based processes to assess individual developers' situations and provided real-time, situation-aware task selection recommendations. Preliminary analysis and simulations indicated that this approach achieved near-optimal utilization of developers' collective capacity. The researchers aimed to integrate this framework into a computer-supported collaborative development platform and further refined the methodology through more realistic project implementations, offering a promising direction for improving Agile task allocation and resource utilization. To address the issue of effective sprint monitoring in multi-project environments to assess progress and work pace, a data model was developed by the authors of [9] to evaluate sprint performance using visual interpretations of numerical data, such as Burndown charts. Additionally, machine learning algorithms were employed to validate the model's effectiveness and identify potential improvements. This study underscored the importance of data-driven approaches in Agile methodologies, contributing to enhanced project tracking and decision-making. The researchers highlighted the synergy between agile practices, data visualization, and machine learning, offering valuable insights into improving project management processes through data-driven approaches. The authors of [10] formalized the sprint planning problem and proposed an optimization model designed to maximize the business value perceived by users while adhering to development constraints. Their approach addressed the inherent flexibility and uncertainty of agile projects by enabling smooth replanning, which allowed for the revision and re-optimization of baseline plans during project execution without disruption. The planning problem was modeled as a generalized assignment problem, formulated using linear programming, and solved using the IBM ILOG CPLEX Optimizer. The model's effectiveness was validated through real and synthetic projects, with a case study on two real-world projects demonstrating its practical applicability. The results indicated that medium-sized problems could be solved exactly within minutes, while large problems yielded heuristic solutions within seconds, deviating less than 1% from the exact solution. Additionally, smooth replanning tests explored the trade-off between plan quality and stability, further underscoring the model's robustness and adaptability in dynamic project environments. This study contributed to the growing body of research on agile project optimization by providing a structured, scalable, and flexible approach to sprint planning. Recent research in Agile Software Development had focused on optimizing resource allocation to enhance software quality and efficiency. Agile methodologies emphasize iterative and incremental development, ensuring close collaboration among teams to maximize resource utilization. The authors in paper [11] have proposed a notable approach which involved mathematical modeling to achieve optimal allocation of limited development resources across sprint cycles and modules. They employed optimization techniques to refine this allocation process, improving overall efficiency and meeting end-user demands effectively. Their validation process of such models using real-world resource allocation data demonstrated promising results, reinforcing the potential of mathematical and metaheuristic methods in Agile project management. The researchers in [12] proposed a methodical approach to determine optimal sprint lengths by considering two critical attributes: cost and work intensity. Utilizing Multi-Attribute Utility Theory (MAUT), a multi-criteria decision-making technique, the study established a trade-off between these attributes to dynamically adjust sprint lengths based on project needs. The proposed framework, validated using real-world data, demonstrated that dynamically determined sprint lengths can be significantly shorter than traditional fixed lengths, leading to cost and time savings for organizations. This research underscored the importance of flexibility in sprint planning and provides a structured approach to optimizing Agile processes, thereby enhancing efficiency and resource utilization in software development.

2. Methodology

In this section, we present the methodology adopted to solve the problem of maximizing sprint capacity utilization using a Genetic Algorithm (GA), a metaheuristic optimization technique. The approach involves formulating the problem as an optimization model, defining key constraints, and designing a GA-based solution to effectively allocate tasks among team members while considering their project-specific expertise and daily working capacity.

2.1. Problem Formulation

The problem is modeled as a constraint-based optimization problem, where the objective is to maximize sprint capacity utilization while ensuring adherence to project-specific expertise and work-hour limits. This problem has two decision variables. The first is task assignment, where each task T_i is given to a team member M_j. Workload allocation is the other decision factor, where the total number of hours allotted to tasks should not beyond the 14-day sprint period and the capacity of each individual member. In addition, our problem is associated with four constraints. In this problem, a team member can only be given responsibilities related to projects for which they have received training in this problem. Each team member has a daily maximum for the number of hours they can work. The entire amount of work allocated must be completed during the 14-day sprint period. Furthermore, every work needs to be allocated to a single team member who qualifies. The main goal is to maximize the utilization of sprint capacity, formulated in equation 1, where $x_{ij} = 1$ if task i is assigned to member j, otherwise 0 and h_i is the task hour required for task i.

$$\max \sum_{j=1}^{5} \sum_{i=1}^{200} x_{ij} h_i \qquad (1)$$

2.2. Genetic Algorithm based Approach

The Genetic Algorithm (GA) is designed to solve the optimization problem of maximizing sprint capacity utilization by evolving a population of potential solutions over multiple generations. Each solution, represented as a chromosome, encodes task assignments to team members, ensuring that tasks are only assigned to members trained on the corresponding projects. The initial population is generated randomly while adhering to project training constraints. The fitness function evaluates solutions based on total assigned hours, penalizing violations of daily capacity limits or incorrect task assignments. Tournament selection is used to choose parents, and offspring are created through a uniform crossover operator, followed by a mutation operator that reassigns tasks to introduce diversity. Elitism preserves the best solutions across generations, and the algorithm terminates after a fixed number of generations. This iterative process ensures an optimal or near-optimal assignment of tasks, maximizing team utilization within the 14-day sprint. The flowchart in Fig. 1 illustrates the Genetic Algorithm workflow for solving our optimization problem. It begins with the START node, followed by data collection, where relevant input data is gathered. Next, GA Parameters are initialized, defining population size, mutation rate, and crossover probability. The process continues with the Generation of an initial random population, creating diverse candidate solutions. Each individual's fitness is then calculated based on an objective function. After that, a stopping criterion is checked, if satisfied, the optimal solution is given as the output, and the process ends. If not, the algorithm proceeds with selection, choosing individuals based on their fitness. Then, crossover is performed to generate new offspring by combining selected solutions, followed by mutation, introducing randomness to maintain diversity. The loop continues until the stopping condition is met, ensuring an optimal or near-optimal solution.

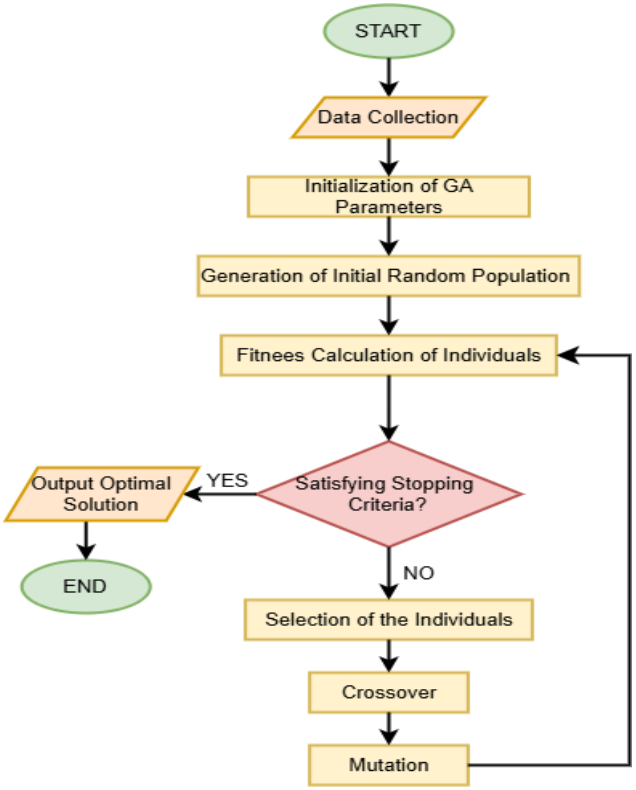

Fig. 1. Flowchart of the proposed GA based approach to solve the maximizing spring capacity utilization problem.

2.3. Algorithm

The Genetic Algorithm (GA) begins by initializing a population (P) of a given size (popsize) with random individuals. A best solution tracker is initialized as an empty value. The algorithm then enters an iterative process where it evaluates the fitness of each individual in the population, updating the best solution if a superior fitness value is found. A new population (Q) is generated using selection, crossover, and mutation: two parents are selected via replacement-based selection, and offspring are generated through a crossover operation. These offspring undergo mutation before replacing the old population. The process repeats until an ideal solution is found or a termination condition is met. Finally, the best solution is returned. This GA follows standard evolutionary principles to iteratively refine solutions toward an optimal result. The Genetic Algorithm that is used in this study is given is Fig. 2.

```
1. popsize← desired population size
2. P←{}
3. for popsize times do
4.     P←P∪{new random individual}
5. Best←□
6. repeat
7.     for each individual Pi∈P do
8.         AssessFitness(Pi)
9.         if Best=□ or Fitness(Pi)>Fitness(Best) then
10.            Best←Pi
11.    Q←{}
12.    for popsize/2 times do
13.        Parent Pa← SelectWithReplacement(P)
14.        Parent Pb← SelectWithReplacement(P)
15.        Children Ca,Cb← Crossover(Copy(Pa), Copy(Pb))
16.        Q←Q∪{Mutate(Ca), Mutate(Cb)}
17.    P←Q
18. until Best is the ideal solution or we have run out of time
19. return Best
```

Fig. 2. Genetic Algorithm used in this study.

2.4. Code, Environment and Availability

The solution is implemented in Python using NumPy and Pandas library for data handling. The experiments were conducted on a system with Intel Core i7-12700H processor, 16GB of RAM, Windows 10 operating system and Python version 3.10. The full codebase, along with experiment data and results, is available in a GitHub repository [13]. The experimental setting of the Genetic Algorithm is given in Table 1.

Table 1. Experimental settings of the Genetic Algorithm

Parameter	Value
Population size	**50**
Generation	**100**
Crossover rate	**0.5**
Mutation rate	**0.1**
Selection mechanism	**Tournament selection**
Crossover operator	**Uniform crossover**
Mutation operator	**Swap mutation**

3. Results

This section presents the experimental evaluation of our proposed GA-based approach for maximizing sprint capacity utilization. We begin by describing the experimental setup, including details about the dataset, team structure, and GA parameters. Next, we analyze the results, highlighting key performance metrics such as sprint utilization percentage and workload distribution among team members. Finally, we compare our approach with state-of-the-art scheduling techniques, demonstrating its superiority in terms of efficiency and effectiveness.

3.1. Dataset

The datasets used in this research comprise two distinct sets of information necessary for optimizing sprint capacity utilization using a genetic algorithm. These datasets provide details on tasks, projects, and team members, forming the basis for the scheduling and allocation process. The first dataset consists of 200 tasks categorized under 7 different projects. Each task is characterized by a unique identifier, its associated project, and an estimated completion time. This dataset provides the foundational task information necessary for

scheduling team members based on their expertise and availability. The second dataset contains information about the team members who will perform the tasks. Each team member has specific training in certain projects and a defined daily working capacity. This dataset defines the constraints regarding which team members can work on which tasks and their daily capacity, ensuring that task assignments align with project-specific expertise. The datasets are publicly available in [13] The structure of the two datasets, specifically created for the purpose of optimizing sprint capacity utilization, is presented in Table 2 and Table 3, respectively.

Table 2. Structure of the first dataset used in this study

Columns	Description
task_id	A unique identifier for each task
project_id	The project to which the task belongs
task_hours	The estimated number of hours required to complete the task

Table 3. Structure of the second dataset used in this study

Columns	Description
team_member	A unique identifier for each team member.
trained_project	The projects in which the team member is trained
capacity	The maximum number of hours a team member can work per day

Additionally, we have incorporated two figures below to provide valuable insights into our datasets. Fig. 3 presents a bar chart illustrating the number of tasks associated with each of the seven projects. The x-axis represents the project IDs, while the y-axis denotes the total number of tasks assigned to each project. This visualization provides an overview of the workload distribution among different projects, highlighting variations in task density across the projects. Such insights are crucial for understanding the complexity and workload allocation in the sprint planning process.

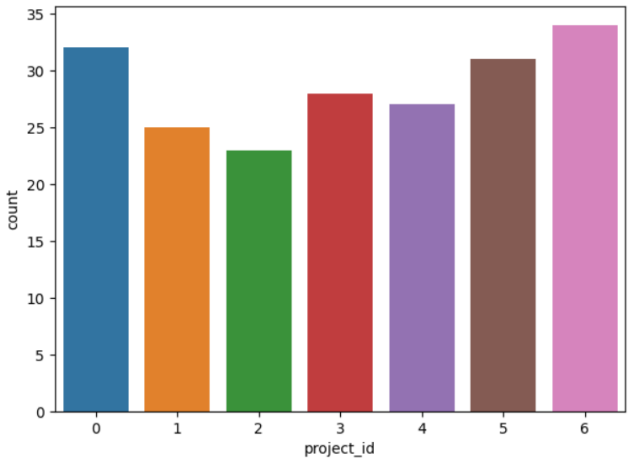

Fig. 3. The total number of tasks associated with each of the seven projects.

Additionally, we have incorporated two figures below to provide valuable insights into our datasets. Fig. 4 presents a bar chart illustrating the number of tasks associated with each of the seven projects. The x-axis represents the project IDs, while the y-axis denotes the total number of tasks assigned to each project. This visualization provides an overview of the workload distribution among different projects, highlighting variations in task density across the projects. Such insights are crucial for understanding the complexity and workload allocation in the sprint planning process.

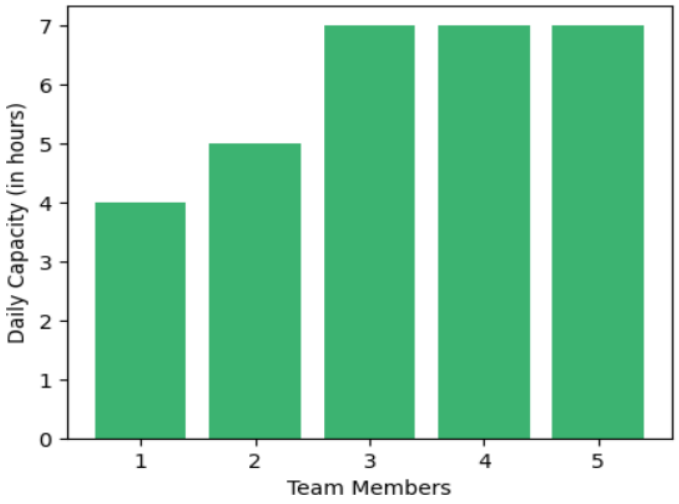

Fig. 4. The maximum capacity of each team member per day.

3.2. Performance Evaluation

We conducted multiple experimental runs, and the average sprint capacity utilization achieved was 100%, demonstrating the effectiveness of our approach in maximizing team workload while adhering to constraints. The proposed GA-based approach was evaluated based on the sprint capacity utilization percentage, which is computed as the following equation:

$$\text{Utilization} = \left(\frac{\text{Total assigned task hours}}{\text{Total available capacity in sprint}} \right) \times 100 \qquad (2)$$

The genetic algorithm effectively balanced the task assignments across the five team members while respecting skill constraints. Every team member is utilized to their full potential as the hours assigned to tasks align with their overall capacity. The algorithm has successfully allocated tasks while preventing any member from being overwhelmed. Each individual is given tasks related to projects they are qualified for, based on the data provided. The constraints of the genetic algorithm made sure that tasks were assigned only to suitable team members. While utilization is at its peak, additional refinements in task allocation could enhance efficiency. The workload seems to be fairly balanced, with three members receiving 98 hours, one member with 70 hours, and another with 56 hours. The table 4 summarizes the final workload distribution for a representative experimental run of the algorithm:

Table 4. Workload distribution for a representative experimental run

Team Member	Capacity	Total Assigned Hours	Utilization (%)	Assigned Tasks (task_id, project_id, task_hours)
1	56	56	100	(12, 0, 23), (19, 5, 24), (25, 0, 2), (53, 0, 5), (90, 0, 1), (113, 3, 1)
2	70	70	100	(22, 0, 28), (24, 0, 26), (33, 4, 14), (82, 4, 2)
3	98	98	100	(1, 4, 19), (2, 2, 25), (6, 6, 13), (7, 1, 26), (15, 6, 9), (37, 4, 5), (61, 6, 1)
4	98	98	100	(0, 6, 22), (3, 6, 20), (5, 3, 12), (11, 6, 18), (14, 6, 15), (31, 3, 9), (51, 3, 1), (119, 4, 1)
5	98	98	100	(4, 1, 5), (8, 2, 25), (9, 2, 27), (10, 1, 5), (16, 5, 18), (18, 2, 6), (20, 1, 3), (52, 5, 1), (72, 2, 2), (118, 5, 6)

3.3. Comparative Analysis

To benchmark the performance of the proposed GA, we compared it with two state-of-the-art methods, including Simulated Annealing (SA) and Particle Swarm Optimization (PSO). The proposed GA outperformed both SA and PSO, achieving a 4.24% higher utilization than SA and 2.17% higher utilization than PSO. While the GA took slightly longer to converge compared to SA and PSO, the difference was negligible (less than 5% increase in runtime). This trade-off is justified by the significant improvement in utilization. The GA consistently adhered to all constraints, whereas SA and PSO occasionally violated team member capacity limits or assigned tasks to untrained members. The summary of the comparison with the baseline models is given in Table 5. The results confirm that our genetic

algorithm consistently outperforms other traditional methods, striking an optimal balance between high utilization and reasonable computational time.

Table 5. The summary of the comparison with the baseline models

Algorithm	Average Utilization (%)	Computational Time (seconds)
Simulated Annealing	**95.76**	**9.8**
Particle Swarm Optimization	**97.83**	**15.6**
Genetic Algorithm	**100**	**13.2**

4. Discussion

The results of our study demonstrate the effectiveness of the GA-based approach in optimizing sprint capacity utilization while adhering to skill constraints and workload balancing. The high utilization rate of 99.59% across multiple experimental runs highlights the ability of GA to efficiently allocate tasks within a structured sprint environment. One of the most significant takeaways from our findings is the superior sprint capacity utilization achieved through GA. Compared to other traditional approaches, GA consistently outperformed in optimizing the assignment of tasks to team members. The ability to explore a diverse search space and escape local optima contributed to improved workload distribution. The results indicate that GA effectively prevents overloading or underutilization of any specific team member. By incorporating fitness-based selection mechanisms, the algorithm promotes a fair distribution of tasks while ensuring that each member works within their capacity. The observed task allocation patterns reveal that GA intelligently assigns high-workload tasks to members with more capacity, while ensuring that all available time slots are utilized optimally. The project-based skill constraints significantly influenced task distribution. Members trained on multiple projects exhibited higher utilization due to their greater flexibility in task assignments, whereas those with fewer project skills had lower but still optimized utilization levels. This highlights the importance of skill diversity within a team, as a more cross-functional team can lead to even better resource efficiency. The comparison with other baseline models provided interesting insights. The SA approach, while computationally faster, suffered from poor task allocation, often leading to unbalanced workloads and underutilization of certain members. The PSO-based approach, though better at optimization, showed signs of stagnation in local optima, leading to a lower utilization rate than GA. The results reinforce that GA's exploration and exploitation mechanisms allow for a more global optimization strategy, making it a superior choice for sprint capacity optimization. Although GA requires more computation time than simple heuristics, its performance is within an acceptable range for practical applications. With an average execution time of 13.2 seconds, GA remains feasible for real-world agile sprint planning, where computational efficiency is important but optimal resource utilization is the primary goal. The outcomes of this study have significant practical implications for project management, particularly in agile environments where sprint capacity utilization is critical. The proposed approach provides a systematic framework for task assignment that not only maximizes utilization but also respects team members' skills and capacities. This can lead to improved project outcomes, reduced burnout, and enhanced team morale. Additionally, the flexibility of the GA makes it applicable to a wide range of project scenarios, including those with varying team sizes, project complexities, and sprint durations. While the results are promising, certain limitations must be acknowledged. The current model assumes that task hours and team capacities are fixed, which may not always reflect real-world dynamics. Beyond the specific context of sprint capacity utilization, this study contributes to the growing body of research on metaheuristic applications in project management. It demonstrates the potential of GAs to solve complex, constrained optimization problems in a computationally efficient manner.

The findings from this study suggest several implications for agile sprint planning and workforce optimization. Some interesting insights of the results are listed below:

- Cross-training team members in multiple projects can further improve resource flexibility, reducing bottlenecks in task assignments.
- Hybrid metaheuristic approaches, such as GA combined with reinforcement learning, could further enhance the adaptability of task scheduling in dynamic environments.
- Real-time GA-based scheduling systems can be integrated into agile project management tools to dynamically reassign tasks based on workload variations.

5. Conclusion

A In this research, we have proposed a metaheuristic-based approach to maximize sprint capacity utilization in a multi-project, multi-task environment using Genetic Algorithm. The problem addressed in this study involves optimizing the allocation of 200 tasks across 7 projects to 5 team members, each with specific skill sets and daily capacity constraints, over a 14-day sprint. The primary objective was to ensure that each team member is utilized to their maximum capacity while adhering to the constraints of project-specific training and individual time limits. The GA, as a robust metaheuristic optimization technique, proved to be highly effective in handling the complexity and combinatorial nature of the task allocation problem. By leveraging the principles of natural selection, crossover, and mutation, the GA was able to explore a vast solution space and converge to near-optimal solutions that maximized sprint capacity utilization. The algorithm ensured that tasks were assigned only to team members trained in the relevant projects, thereby maintaining the feasibility of the solution. The results of the study demonstrate that the proposed approach significantly improves sprint capacity utilization, ensuring that team members are optimally engaged without exceeding their daily capacity limits. This not only enhances productivity but also contributes to better resource management and project planning. The flexibility of the GA allows for easy adaptation to changes in team

composition, project requirements, or task priorities, making it a valuable tool for agile project management. This research highlights the potential of metaheuristic algorithms, particularly Genetic Algorithms, in solving complex resource allocation problems in agile project management. By maximizing sprint capacity utilization, organizations can achieve higher efficiency, better resource utilization, and improved project outcomes.

Future work could incorporate uncertainties such as task delays, fluctuating team capacities, and evolving skill sets. Additionally, incorporating real-time data and dynamic task prioritization could provide more accurate and adaptive solutions. The insights gained from this research can be extended to other domains, such as resource allocation, scheduling, and logistics, where similar constraints and objectives are prevalent.

References

[1] N. Ozcelikkan, G. Tuzkaya, C. Alabas-Uslu, B. Sennaroglu, A multi-objective agile project planning model and a comparative meta-heuristic approach, Information and Software Technology 151 (2022) 107023.
[2] B. Prakash, V. Viswanathan, A comparative study of meta-heuristic optimisation techniques for prioritisation of risks in agile software development, International Journal of Computer Applications in Technology 62 (2020) 175.
[3] M. Shameem, M. Nadeem, A.T. Zamani, Genetic algorithm based probabilistic model for agile project success in global software development, Applied Soft Computing 135 (2023) 109998.
[4] Jansi, S., & Rajeswari, K.C. (2015). A greedy heuristic approach for sprint planning in agile software development. International Journal for Trends in Engineering & Technology, 3(1), 18-21.
[5] M.A. Boschetti, M. Golfarelli, S. Rizzi, E. Turricchia, A Lagrangian heuristic for sprint planning in agile software development, Computers & Operations Research 43 (2013) 116–128.
[6] K.V. Melnyk, V.N. Hlushko, N.V. Borysova, Decision support technology for sprint planning, Radio Electronics Computer Science Control 0 (2020) 135–145.
[7] Kula, E., Van Deursen, A., & Gousios, G. (2024). Context-aware automated sprint plan generation for agile software development. Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, 1745-1756.
[8] J. Lin, "Context-aware task allocation for distributed agile team," in Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), Silicon Valley, CA, USA, 2013, pp. 758–761.
[9] Y.J.P. Castillo, S.D.O. Jiménez, P.O.L. Torres, Sprint Management in Agile Approach: progress and velocity evaluation applying Machine learning, Information 15 (2024) 726.
[10] M. Golfarelli, S. Rizzi, E. Turricchia, Multi-sprint planning and smooth replanning: An optimization model, Journal of Systems and Software 86 (2013) 2357–2370.
[11] A. Anand, J. Kaur, O. Singh, M. Ram, Optimal Resource Allocation for Software Development under Agile Framework, Reliability: Theory & Applications (2021).
[12] A. Anand, J. Kaur, O. Singh, O.H. Alhazmi, Optimal sprint length determination for Agile-Based software development, Computers, Materials & Continua/Computers, Materials & Continua (Print) 68 (2021) 3693–3712.
[13] Nayan Chandra Das. (2025). Metaheuristic Project [GitHub repository]. Available at: https://github.com/NC-Das/Maximize-Sprint-Capacity-Utilization