Received: xxx **Accepted:** xxx **Published:** xxx.
DOI. xxxxxxx
xxx Vol. xxx, No. xxx, (1-11)

Research Article

Open Access

Control and Optimization in Applied Mathematics - COAM

Determining Control Points in the Project Life Cycle: A Heuristic Approach Utilizing Tabu Search

Narjes Sabeghi ✉

Department of Mathematics, Faculty of Basic Sciences, Velayat University, Iranshahr, Iran.

✉ **Correspondence:**
Narjes Sabeghi
E-mail:
n.sabeghi@velayat.ac.ir

Abstract. A critical aspect of successful project management is ensuring that execution aligns with the baseline schedule. However, traditional project control methods often struggle to effectively address the uncertainties and deviations that can arise during project execution, leading to delays and inefficiencies. To tackle these challenges, this paper introduces a novel heuristic approach based on the Tabu Search (TS) algorithm for identifying discrete control points throughout the project life cycle. These control points enable proactive monitoring, timely deviation detection, and corrective actions, significantly minimizing project delays. Unlike traditional scheduling techniques, which can be rigid and reactive, our proposed method dynamically adjusts control points to enhance project oversight. Experimental results on benchmark instances from the Kolisch library demonstrate that our approach significantly reduces project delays, with up to 20% improvements compared to initial schedules in certain scenarios. These findings underscore the effectiveness of the TS algorithm in enhancing project control strategies, highlighting its potential applicability in real-world project management scenarios.

How to Cite
Sabeghi, N. (2025). "Determining control points in the project life cycle: A heuristic approach utilizing tabu search", Control and Optimization in Applied Mathematics, 10(): 1-11, doi: 10.30473/coam.2025.73812.1290

Keywords. Project management, Scheduling, Proactive management, Control points, Tabu search algorithm.

MSC. 90B50; 90C59.

1 Introduction

In the management of complex projects, maintaining alignment with the baseline schedule is critical to ensuring successful project delivery. However, due to uncertainties, resource constraints, and unforeseen events, deviations from the initial schedule often occur. These deviations, if not addressed promptly, can lead to delays, budget overruns, and suboptimal outcomes. Therefore, identifying discrete control points throughout the project life cycle where progress can be assessed and corrective actions can be taken becomes a vital aspect of project management.

Historically, project control relied heavily on deterministic approaches like the Critical Path Method (CPM), Program Evaluation and Review Technique (PERT), and Gantt charts. These tools, while foundational, have limitations due to their inability to manage uncertainties and risks in project execution. As noted by Pellerin and Perrier (2018), traditional methods are largely schedule-driven and focus on activities' precedence relationships, without fully considering the stochastic nature of real-world projects. The simplifications made in these techniques often compromise their applicability to complex projects where uncertainties play a significant role [4].

Resource-constrained project scheduling problems (RCPSP) involve allocating a set of limited resources to various activities within a project, while considering precedence constraints -which dictate the order in which activities must be completed- and specific project objectives es) in [9]. Effective project scheduling is crucial for ensuring timely and resource-efficient project completion. The RCPSP represents a well-established combinatorial optimization challenge that arises in complex projects where both resources limitation and task dependencies are crucial. Numerous methods have been explored to address the RCPSP, ranging from traditional heuristics to advanced metaheuristic algorithms such as Tabu Search (TS) (e.g., [1, 8]), Genetic Algorithms (GA), and Simulated Annealing (SA). These methods aim to find a near optimal resource allocation while minimizing project duration and meeting predefined constraints [3].

Although many advanced project scheduling algorithms have been developed to construct resource-feasible baseline schedules, projects are often executed in relatively unstable environments, making changes to the baseline schedules inevitable [6]. Therefore, monitoring and control of project progress are essential. One objective of the control process may be to minimize the deviations from the baseline schedule (e.g., refer to [5, 6]). Despite significant efforts to make project schedules more robust to unforeseen events and the many solutions proposed by researchers, a comprehensive monitoring and control mechanism throughout project execution remains essential. Unlike exact optimization approaches, such as the facility location model proposed by Sabeghi et al. (2015) [6], which provides mathematically rigorous solutions, our study focuses on a heuristic-based approach. The use of TS offers greater flexibility and computational efficiency, particularly in large-scale and dynamic project environments where exact methods may become computationally expensive.

To gain a clearer insight into the current research landscape surrounding project control, Song et al. in [7] analyzed and classified existing studies into four primary categories: (1) project monitoring methods, (2) corrective action strategies, (3) modelling constraints, and (4) the use of project data. In the category of project monitoring, two main approaches are discussed: top-down and bottom-up monitoring. Top-down project monitoring evaluates overall project performance against predefined threshold values and generates warning signals when these thresholds are exceeded, indicating potential project delays. In contrast, bottom-up monitoring begins at the activity level, measuring the progress of individual tasks and comparing them to activity-specific thresholds to detect early warning signs.

Among the various strategies for top-down monitoring, the literature identifies three primary approaches: Buffer Management (BM), Earned Value Management (EVM), and Statistical Analysis (STA). BM focuses on monitoring buffer consumption against expected values; EVM uses metrics such as Schedule Performance Index (SPI) and the Cost Performance Index (CPI) and their time-based counterparts to assess schedule and cost performance; and STA employs methods like traditional statistical techniques, control charts, Bayesian statistics, and Kalman filters to detect deviations and define threshold limits. Most of the existing research, as summarized in Song et al. [7] emphasizes EVM as a dominant tool for top-down project monitoring.

The approach proposed in our study also falls within the category of top-down project monitoring. Specifically, discrete control points are determined across the project life cycle using the TS algorithm. The project performance is then simulated to identify potential delays. If the simulated project completion time exceeds the planned deadline, corrective actions are taken. These corrective actions consist of crashing compressible activities within their feasible limits. In this study, we do not consider resource constraints in the crashing process. Moreover, control actions are implemented only at the end of activity execution, and we assume no activity preemption occurs.

The use of TS in this study is motivated by its ability to dynamically adjust its exploration strategy through adap tive memory structures. Unlike GA and SA, which depend on randomized mechanisms for diversification, TS systematically restricts revisits to previously explored solutions, thereby enhancing search efficiency and improving convergence toward high-quality solutions.

The remainder of the paper is organized as follows: Section 2 provides a detailed explanation of the proposed TS algorithm for determining project control points (DPCP). In Section 3, the results of implementing this algorithm on several test problems from the Kolisch library are presented. Finally, Section 4 offers the discussion and conclusion.

2 Tabu Search Algorithm for Determining the Project Control Points

In this section we implement a TS algorithm to identify discrete control points in a project life cycle. The goal is to track and control the progress of the project by finding the near optimal placement of control points, which allows for monitoring, evaluation, and corrective action during project execution. The details of the presented TS algorithm are outlined in Algorithm 1 and for convenience, a flowchart illustrating this algorithm is provided in Figure 1.

This algorithm identifies all paths in the project network, from the starting node to the end node. These paths are important because they determine the critical sequences of activities that influence the overall project time line.

An initial random solution is generated by selecting a random subset of the control points (cp) from the set of existing points, Ep. Ep is the set of all discrete time instants where the execution of one or more activities of the project finishes according to the baseline schedule. cp is the sequence of the points which represent the initial set of control points where project progress will be monitored. The initial cost of this solution is evaluated using the cost function, $costfunc()$, introduced by Algorithm 2. This function, estimates the total penalty of a given solution by simulating project progress and calculating the average delay with respect to the project due date. It takes into account possible variations in activity durations, evaluates critical and non-critical paths, and quantifies how delays accumulate under different control point configurations. First, input variables, such as initial start times and durations of activities, are initialized, and the critical and non-critical project paths are identified. For each activity, a random duration is generated within a specified range, which is $[minval, maxval]$ ($minval$: Minimum possible durations for activities (activities' compressed duration), $maxval$: Maximum possible durations for activities) and project finish times are updated. For each control point, the critical activities duration from that control point to the end are reduced, and the impact on the project's final completion time is evaluated. Finally, if the project exceeds the due date, a penalty is calculated as a percentage of the relative delay, and this value is returned as the total cost:

$$C_{\max} = ((pft(n) - duedate)/duedate)/Maxsii \times 100,$$

where, $pft(n)$ is the project finish time and $Maxsii$ is the maximum number of simulation iteration.

The TS process is implemented to iteratively improve the initial solution by exploring neighbouring solutions and avoiding previously explored regions using a Tabu List, TL. TL is a matrix that is used to keep track of moves that are temporarily forbidden, preventing the algorithm from revisiting recently explored solutions.

In each iteration, a neighbouring solution is generated by randomly modifying one of the control points. The algorithm explores alternative positions for this control point by moving

Algorithm 1 TS Algorithm for DPCP.

Input:

- **Project Information:**
  - $st_{initial}$: Array of initial start times for activities.
  - $d_{initial}$: Array of activity durations.
  - n: The number of project's activities.
  - pre: An array containing the list of predecessor activities for each activity.
  - n_{prec}: An array indicating the number of predecessors for each activity.
  - $succ$: An array containing the list of successor activities for each activity.
  - n_{succ}: An array indicating the number of successors for each activity.
  - pft: Activities' finish times.
  - m: The number of control points.
- **TS Parameters:**
  - $MaxIt$: Maximum number of iterations.
  - lTL: Tabu tenure.

Output: Relative delay of the project after completing execution within the control process.

Procedure:

1. Identify all possible project paths.
2. Generate initial control points (Ep):
   (a) Identify control points (Ep) from finish times pft. These points are potential control points.
   (b) Sort Ep and select m control points (cp) randomly from Ep.
   (c) Evaluate the cost of the initial solution using $costfunc()$ (refer to Algorithm 2) and set it as the best solution $BestSol$.

it both forward (to later positions in Ep) and backward (to earlier positions in Ep) within the ordered set of candidate control points (potential control points). Each modified solution is evaluated using the $costfunc()$. If the new solution offers an improvement over the current best solution and is not restricted by the Tabu List, it is adopted as the new best solution. Subsequently, The TL is updated to record the recent moves, ensuring they are not revisited within a short period (Tabu tenure).

The algorithm continues iterating until it reaches the maximum number of iterations ($MaxIt$). After each iteration, the best solution found so far is stored. The final solution represents the best-found placement of control points for project monitoring.

Algorithm 2 Cost Function Calculation ($costfunc()$)

```
Input:
  • $sol$: Current solution (control points).
  • $n$: Number of activities.
  • $st_{initial}$: Initial start times.
  • $d_{initial}$: Initial durations.
  • $n_{prec}, n_{succ}$: Predecessors and successors.
  • $duedate$: Project due date.
  • $\Pi$: Project paths.
  • $minval, maxval$: Minimum and maximum durations for activities.
  • $Maxsii$: Maximum simulation iterations.
Output: $C_{\text{max}}$: Total cost based on project delay.
Procedure:
  1. Initialize:
       • Set $fstatus$ and $sstatus$ to 0;
       • Initialize start times ($st$), durations ($d$), and finish times ($pft$).
  2. For each simulation iteration $sii$ from 1 to $Maxsii$:
       (a) Generate a random duration $reald(i)$ in $[minval(i), maxval(i)]$.
       (b) Update $realft(i)$ and start times for successor activities.
       (c) Calculate total duration $lp(i)$ for each path in $\Pi$.
       (d) Identify critical ($cp$) and non-critical paths ($np$).
       (e) Compute possible reduction in critical activity durations ($maxcrashing$).
       (f) Simulate progress by adjusting activity durations ($reald$) and recalculate $pft$.
       (g) If $pft(n) > duedate$, calculate the relative delay as a percentage.
  3. Compute mean relative delay: $C_{\text{max}}$.
  4. Return $C_{\text{max}}$.
```

In Algorithm 2, The corrective actions are modelled through activity crashing at control points. At each selected control point, the algorithm identifies critical activities and applies duration reductions within their feasible ranges. This reflects managerial decisions made during project monitoring to mitigate delays. Other corrective actions may include adding resources or adjusting task sequencing.

In our approach, the corrective actions related to modifying control points involve dynamic simulation of project progress and adaptive duration reduction (crashing) of critical activities. Specifically, each time a new set of control points is generated, the algorithm performs the following:

- It simulates the project execution up to each control point,
- Identifies the current critical paths based on updated durations,
- Evaluates the potential to reduce the durations of critical activities at those points (based on predefined flexibility bounds),
- Applies a crashing mechanism to simulate corrective actions (e.g., allocating extra resources or compressing schedules),
- Recalculates the project finish time accordingly.

This allows the model to determine how effective a particular set of control points would be in reducing the relative project delay. The TS process then seeks a configuration of control points that maximizes the impact of corrective actions, leading to a lower total project delay.

3 Experimental Results

Algorithms 1 were implemented in MATLAB and tested on several benchmark instances from the Kolisch library [2], which provides standard test cases for project scheduling. These benchmark problems, such as the J30 problem set, contain varying task durations, resource constraints, and dependencies, which make them suitable for testing the efficiency and robustness of the proposed TS algorithm.

In each test, the initial control points were selected based on the baseline schedule to provide starting points for the TS process. Sabeghi et al. [6] made some effort to find the best amount of number of control points. For J30, they found that considering seven control points leads to producing better results. Hence, without loss of generality, we use this quantity in our computations. Table 1 presents the results, comparing the average relative delay (% RD) for initial and improved control points across several instances. For these instances, based on

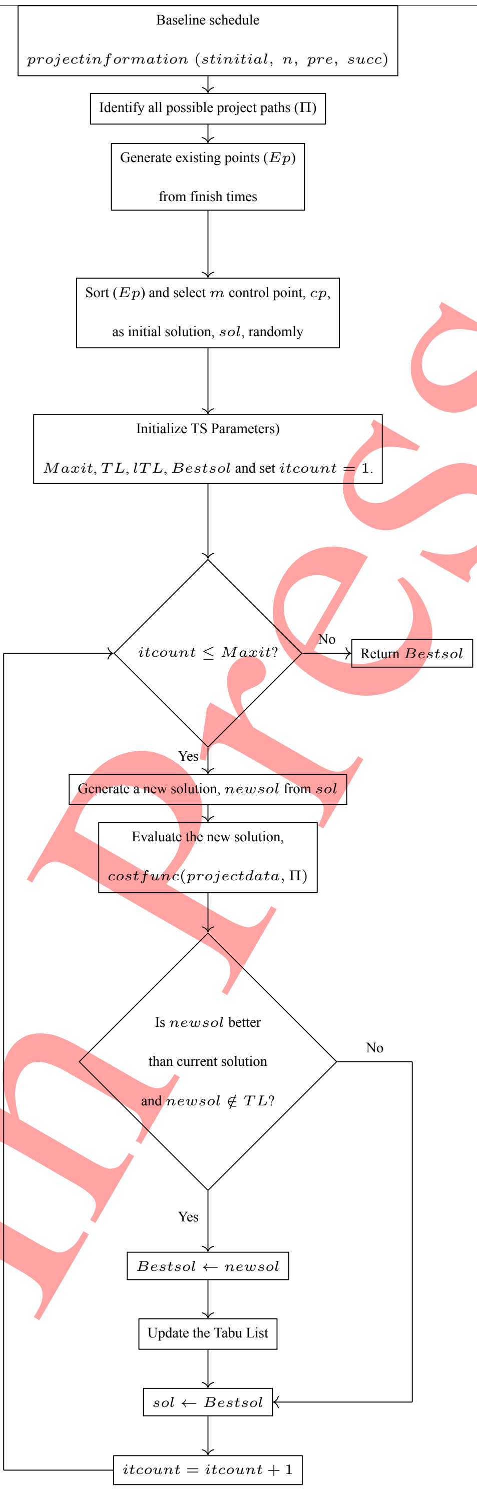
Baseline schedule
$projectinformation\ (stinitial,\ n,\ pre,\ succ)$
Identify all possible project paths (Π)
Generate existing points (Ep)
from finish times
Sort (Ep) and select m control point, cp,
as initial solution, sol, randomly
Initialize TS Parameters)
$Maxit, TL, lTL, Bestsol$ and set $itcount = 1$.
$itcount \leq Maxit?$
No
Return $Bestsol$
Yes
Generate a new solution, $newsol$ from sol
Evaluate the new solution,
$costfunc(projectdata, \Pi)$
Is $newsol$ better
than current solution
and $newsol \notin TL$?
No
Yes
$Bestsol \leftarrow newsol$
Update the Tabu List
$sol \leftarrow Bestsol$
$itcount = itcount + 1$

Figure 1: Outline of the proposed method

preliminary experiments to ensure a balanced trade-off between solution quality and computational efficiency, we selected a Tabu tenure of 10 and a maximum iteration count of 100. As shown in the table, the algorithm effectively reduced the project delay by iteratively adjusting control points to minimize deviation from the baseline schedule.

The process began with generating a set of initial control points and then applying the TS algorithm iteratively to improve these control points. For example, in instance j3010-5, the initial average relative delay of 29.64% was reduced to 9.54% after adjustment, demonstrating an improvement. Similarly, for instance j3035-5, the delay was reduced from 20.27% to 14.62%, showing the algorithm's capability to adjust control points strategically and mitigate delays effectively. In addition to the differences in the characteristics and complexity of the benchmark problems, the observed variations in the performance of the TS algorithm are also influenced by the simulation of the execution process. For each problem instance, the actual execution times of activities are randomly generated, and delays are calculated accordingly. As a result, the probability of experiencing higher or lower delays differs across projects, affecting the feasibility of corrective actions. This inherent variability in the simulation process contributes to the observed differences in the effectiveness of the TS algorithm across various instances. Therefore, the ability of TS to reduce delays depends not only on the structural properties of the benchmark problems but also on the stochastic nature of project execution.

Each test was conducted multiple times to account for variability in project conditions and confirm the robustness of the proposed method. The algorithm's effectiveness was validated by comparing initial and final control points, with improved placements resulting in decreased delays in nearly all instances. These results highlight the adaptability of the TS algorithm, showcasing its potential to be applied broadly to various projects to improve adherence to baseline schedules.

Table 1: The average relative delay (%RD) for some instances of $J30$

File number	Due date	Initial Control Points	%RD	Final Control Points	%RD
j301-5	41	10, 11, 17, 19, 38, 39, 41	7.06	4, 6, 12, 16, 29, 37, 41	3.83
j305-5	87	13, 19, 35, 36, 51, 65, 87	2.30	9, 13, 19, 35, 51, 65, 87	1.54
j3010-5	43	2,15, 20, 21, 31, 38, 43	29.64	2, 9, 15, 20, 31, 38, 43	9.54
j3015-5	65	16, 29, 31, 33, 35, 43, 65	7.63	5, 29, 31, 33, 35, 43, 65	2.99
j3020-5	61	2, 11, 12, 19, 35, 37, 61	21.58	2, 11, 12, 19, 23, 35, 61	16.98
j3025-5	77	16, 32, 39, 55, 65, 66, 77	2.15	8, 32, 39, 55, 65, 66, 77	0.86
j3030-5	58	5, 15, 27, 30, 43, 49, 58	9.36	5, 27, 30, 43, 46, 49, 58	8.22
j3035-5	60	24, 28, 32, 34, 40, 49, 60	20.27	14, 28, 32, 34, 40, 49, 60	14.62
j3040-5	65	12, 26, 27, 37, 49, 55, 65	27.26	12, 26, 27, 31, 37, 49, 65	16.82
j3045-5	97	11, 18, 49, 50, 65, 72, 97	5.40	11, 49, 50, 65, 72, 92, 97	1.55

4 Conclusion

This study presented a Tabu Search (TS) algorithm designed to identify the near-optimal placement of control points throughout the project life cycle, enabling proactive monitoring and timely corrective actions. By systematically adjusting the positioning of these control points, the algorithm effectively reduced project delays across several benchmark instances from the Kolisch library. The results indicate that the algorithm can be readily adapted to various types of projects, offering project managers with a reliable tool for monitoring progress and swiftly addressing deviations from the base-line schedule. Moreover, the flexibility of the TS algorithm paves the way for its implementation in diverse project management contexts, potentially enhancing both efficiency and productivity. Future research could explore integrating additional constraints, such as cost considerations and resource allocation, to further refine the approach. Exploring these dimensions could broaden the algorithm's applicability and effectiveness, offering project managers enhanced strategies for navigating complex project environments and optimizing resource utilization.

Declarations

Availability of Supporting Data

All data generated or analyzed during this study are included in this published paper.

Funding

The authors conducted this research without any funding, grants, or support.

Competing Interests

The authors declare that they have no competing interests relevant to the content of this paper.

References

[1] de Soto, B.G., Rosarius, A., Rieger, J., Chen, Q., Adey, B.T. (2017). "Using a tabu-search algorithm and 4D models to improve construction project schedules", *Procedia Engineering*, 196, 698-705, doi:10.1016/j.proeng.2017.07.236.

[2] Kolisch, R., Sprecher, A. (1997). "PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program", *European Journal of Operational Re-*

search, 96(1), 205-216, doi:10.1016/S0377-2217(96)00170-1.

[3] Pan, N.H., Hsaio, P.W., Chen, K.Y. (2008). "A study of project scheduling optimization using tabu search algorithm", *Engineering Applications of Artificial Intelligence*, 21(7), 1101-1112, doi:10.1016/j.engappai.2007.11.006.

[4] Pellerin, R., Perrier, N. (2019). "A review of methods, techniques and tools for project planning and control", *International Journal of Production Research*, 57(7), 2160-2178, doi:10.1080/00207543.2018.1524168.

[5] Sabeghi, N., Tareghian, H.R. (2020). "Using the generalized maximum covering location model to control a project's progress", *Computational Management Science*, 17(1), 1-21, doi:10.1007/s10287-018-0301-5.

[6] Sabeghi, N., Tareghian, H.R., Demeulemeester, E., Taheri, H. (2015). "Determining the timing of project control points using a facility location model and simulation", *Computers & Operations Research*, 61, 69-80, doi:10.1016/j.cor.2015.03.006.

[7] Song, J., Martens, A., Vanhoucke, M. (2022). "Using earned value management and schedule risk analysis with resource constraints for project control", *European Journal of Operational Research*, 297(2), 451-466, doi:10.1016/j.ejor.2021.05.036.

[8] Thomas, P.R., Salhi, S. (1998). "A tabu search approach for the resource constrained project scheduling problem", *Journal of Heuristics*, 4, 123-139, doi:10.1023/A:1009673512884.

[9] Yang, B., Geunes, J., O'Brien, W.J. (2001). "Resource-constrained project scheduling: Past work and new directions", *Department of Industrial and Systems Engineering, University of Florida, Tech. Rep.*