Journal of Construction Engineering, Management & Innovation
2024 7(3):247-265
DOI 10.31462/jcemi.2024.03247265

RESEARCH ARTICLE

Effects of topological structure of project network on computational cost

Saman Aminbakhsh

Atilim University, Department of Civil Engineering, Ankara, Türkiye

Article History

Received 19 August 2024
Accepted 23 September 2024

Keywords

Multi-objective optimization
Network complexity
Pareto front
Project scheduling
Topological structure

Abstract

Understanding how network complexity affects optimization algorithms is crucial for improving computational efficiency. This study investigates how variations in network complexity impact the performance of optimization algorithms. By examining networks with different serial/parallel indicator (I2) values, the research uncovers several key insights into how topology influences computational requirements. The experiments show that higher I2 values, which are closer to serial configurations, heighten the problem's complexity. This study reveals that networks with lower I2 values, which exhibit steeper time-cost curves with fewer solutions over their efficient frontiers, require significantly more CPU time, indicating that project complexity does not necessarily scale with the extend of the Pareto fronts. This contradicts the expectation that more Pareto front solutions would inherently demand greater computational resources. Lastly, the study highlights that while the number of time-cost realizations is often used to gauge project complexity, it may not be conclusive on its own and that one complexity measure can outperform another. Although it can be an effective indicator, it does not fully capture the computational challenges posed by different network topologies. This study further acknowledges the difficulty in establishing a clear link between project performance and complexity due to the multifaceted nature of the problem. The findings suggest that exploring similar problems in other contexts could provide valuable insights into understanding and managing computational complexity.

1. Introduction

Project scheduling is integral to effective project management, and over time, various strategies and tools have emerged to enhance this process. Among these, the Critical Path Method (CPM), introduced by Kelley and Walker [1], stands out as a fundamental and widely practiced technique. CPM is a method based on network analysis that aids in planning, scheduling, and controlling complex projects. It operates on the premise that when the activity duration amounts are defined, CPM can determine the minimum project duration and identify which tasks and paths are critical. CPM employs the project network for modeling a project, which needs to be a directed, acyclic graph linking the activities, including two milestones denoting start and finish of the project.

Since the project network forms the basis for CPM calculations, the overall effectiveness of this simple scheduling technique significantly relies on

Correspondence Saman Aminbakhsh saman.aminbakhsh@atilim.edu.tr

eISSN 2630-5771

the complexity of the network. This is why the topological structure of the network becomes crucial, and its impact on the complexity and the computational resource requirements of the problem needs to be addressed in detail. A range of studies have therefore focused on the application of CPM within the context of construction project network topology. For instance, Hegazy [2] and Ökmen [3] concentrate on linear construction projects with repetitive activities. Ökmen [3] presents a CPM scheduling procedure while Hegazy [2] integrates CPM with the line of balance technique (LOB) technique in their model. Shankar et al. [4] improve on CPM by introducing an ant colony optimization (ACO) algorithm, demonstrating its improved capabilities in determining the critical and sub-critical paths in the network. Such studies have contributed to a deeper understanding and application of CPM in construction project network topology. Moreover, the complexity of the optimization problem has been recognized and addressed in a range of studies on diverse optimization problems [5-8]. However, given the sparse literature with a specific focus on the application of CPM within the context of network complexity, the overview of body of literature provided herein will not only cover these areas, rather will also focus on the relevant research concentrating on the generation of new test sets and the measurement of perceived complexity in various optimization problems.

The coefficient of network complexity (CNC), for example, was first defined by Pascoe [9] for activity-on-arrow (AoA) networks as the number of arcs over the number of activities which was later modified by Davies [10] and Kaimann [11]. CNC is one of the very earliest and perhaps the best-known metric characterizing the topological structure of project networks. Later, Davis [12] adapted this metric for activity-on-node (AoN) networks as the number of direct arcs over the total number of activities. Mastor [13] present an alternative indicator of the topological structure of AoN project networks known as the order strength (OS). OS is defined as the number of precedence relations excluding the arcs connecting the start and finish milestone activities divided by the theoretical maximum number of precedence relations. OS is another well-known metric used to reflect complexity of a project network.

Elmaghraby and Herroelen [14] address the need for measuring the complexity of activity networks to estimate computing requirements and compare heuristic procedures. This study refers to network complexity as the difficulty in analyzing and synthesizing a network and discusses the relationship between the measures of network complexity (MNC) and the theory of combinatorial complexity. Elmaghraby and Herroelen [14] generate a total of 104 different networks based on the hybrid algorithm of Herroelen and Caestecker [15] where each network is characterized by the number of nodes and arcs as well as a random topological structure. This study introduces a measure of network complexity based on the number of multiplications, convolutions, and integrations of arcs, though the challenges in measuring network complexity with varying resource availability is acknowledged. Liu and Chen [16] extend the concept of the critical path to interval plan networks by introducing the deadline-based interval critical path (DBICP) and presenting an algorithm for its determination.

Bein et al. [17] propose complexity index (CI) for two-terminal acyclic AoA project networks. CI is defined as the reduction complexity, or the smallest number of node reductions that, in addition to series and parallel reductions, permit the reduction of a two-terminal acyclic network to a single edge. CI in essence quantifies how close an AoA network is to a series-parallel directed graph. Kolisch et al. [18] introduced ProGen, a network generator for AoN networks that considers both network topology and resource-related characteristics. Schwindt [19] subsequently expanded ProGen into ProGen/Max, enabling the handling of three distinct types of resource-constrained project scheduling problems, including those with minimal and maximal time lags. Agrawal et al. [20] emphasized the importance of CI indicator and developed DAGEN, an AoA

network generator, which allows for the presetting of this complexity measure.

However, Demeulemeester et al. [21] argued that the networks generated by these tools cannot be considered strongly random, as they do not guarantee that the topology is a random selection from the space of all networks that satisfy the specified input parameters. Demeulemeester et al. [21] criticize Pascoe's [9] CNC metric for its inability to distinguish between simple and complex instances, rendering it inadequate as a measure of how network topology influences project scheduling difficulty. To address these limitations, Demeulemeester et al. [21] propose an instance generator known as RanGen, that creates networks with predetermined CI and OS values and taking into account the pre-specified CPU time-limit and the maximum number of networks. RanGen network generator is designed to implement a recursive enumeration method in order to prevent the generation of networks with identical topological structures. Later, Vanhoucke et al. [22] introduced RanGen2, a modified version of Demeulemeester et al.'s [21] RanGen, incorporating alternative topological indicators. Their findings indicate that RanGen2 outperforms ProGen, RanGen, and RiskNet [23] in terms of the total number of networks generated. Starting from a larger set of possible networks, RanGen2 is shown to be capable of generating multiple distinct networks in a single run.

Vanhoucke and Maenhout [24], debating that there is a growing demand for a base for comparison in the nurse scheduling problem research community, present benchmark scheduling instances. This study also presents complexity indicators, including problem size, to predict complexity of the scheduling problems and as a result the computational effort of the solution procedures. This study explores the one-dimensional and two-dimensional relationships between the indicators and their effects on problem complexity. The paper also presents a regression tree model for predicting problem complexity and discusses the discriminative and predictive power of the proposed complexity indicators. Based on the proposed complexity indicators, a tool named NSPGen is presented for generating benchmark instances which have been used to evaluate the behavior of meta-heuristics under different parameter values for the complexity indicators.

Batselier and Vanhoucke [25] by focusing on the project control phase and Earned Value Management (EVM), introduce the concept of project regularity which measures the deviation of the project's planned value curve from a perfectly linear curve and suggest that project regularity could be more influential than project seriality (expressed by the serial/parallel-indicator, SP or I2). The paper suggests that project regularity could be useful for project network generators. Project regularity which categorizes projects as strongly irregular, mildly irregular, or regular is a new characteristic that reflects the value accrued within a project, expressed in terms of the regular/irregular-indicator (RI). Results of this study show that project irregularity has an adverse effect on both time and cost forecasting accuracy. Ellinas et al. [26] address the challenge of quantitatively assessing project complexity by using empirical activity networks to measure the technological aspect of five real-world projects. The study presents a procedure that uses activity networks to capture structural complexity in a quantitative manner. This study highlights that project complexity does not necessarily scale with project cost or size and that the similarities between activity networks and complex systems, such as the internet, are indisputable.

Van Den Eeckhout et al. [27] address the project staffing problem with discrete time-resource trade-offs to minimize personnel staffing budget. This study embeds activity scheduling flexibility by incorporating the project scheduling problem and introduces extra demand scheduling flexibility via discrete time-resource trade-offs. The paper discussing the significance of network topology during planning, utilizes the multi-mode PSPLib dataset to generate various network topologies for problem instances based on different complexity indicators. This research leverages the ProGen network generator for generation of problem

instances with duration and resource demand characteristics set based on ProGen and RanGen project generators. The provision of data for a realistic case is undoubtedly crucial and highly beneficial. In this regard, aiming to provide a test bed for examining hypotheses, algorithms, Probabilistic Risk Modeling and Decision outcomes models, Thiele et al. [28] present a dataset containing a portfolio comprising six projects across which a set of interrelated resources are utilized.

Queiroz et al. [29] discuss that the size influences the challenge during experiments, affecting computational time and solution quality. In this respect, Queiroz et al. [29] introduce REQreate, a tool for generating realistic instances for on-demand transportation problems such as the Dial-a-Ride Problem (DARP) and On-demand Bus Routing Problem (ODBRP). The tool addresses the common practice of using instances based on artificial networks or specific city data by providing a more flexible and diverse approach for generating instances. REQreate tackles the lack of standard benchmark sets by generating instances based on real-world networks from OpenStreetMaps and assists in testing optimization algorithms. Since this tool can be configured to generate instances with different sizes, levels of urgency, dynamism, and geographic dispersion, their effect on performance metrics for the Meal Delivery Routing Problem (MDRP) is evaluated. Deploying bipartite graph and maximal matching, this paper also introduces the concept of instance similarity for informative algorithm analyses.

Coelho and Vanhoucke [30] introduce a new complexity indicator to measure the difficulty of project instances and propose a new data generation method for creating challenging instances of the resource-constrained project scheduling problem (RCPSP). This study proposes numerous instances, notably a new set of 390 hard project instances generated based on a modified version of the "going to the core" algorithm. By analyzing the solution space of 10,793 instances, this study presents an instance hardness measurement named as Sigma distance method based on the average and standard deviation of the solution space and a chosen reference point. Snauwaert and Vanhoucke [31] further contributes by proposing a new data generation procedure as well as artificial datasets for multi-skilled resource-constrained project scheduling problem (MSR-CPSP) using which the hardness of the multi-skilled project instances is investigated. The authors generate 500 networks including 30 to 90 activities with SP indicator values as 0.1, 0.3, 0.5, 0.7, and 0.9 by the RanGen2 generator incorporating skill and resource constraints. To evaluate the dataset, they use scatterplots to analyze the feature space and compare it to existing benchmark datasets. The computational experiments reveal that the percentage of optimal and feasible solutions for mixed integer linear programming (MILP) model increases with increasing SP values and that the number of optimal and feasible solutions decreases with an increasing number of activities. The results also showed that the serial-parallel network indicator value SP has a high impact on the hardness of the instances.

Snauwaert et al. [32] formally define and explain various multi-skilled workforce formation problems, analyze their complexity, and outline the structure of the complexity proofs. This study addresses multi-skilled project scheduling and workforce scheduling, examining genetic algorithms (GA), simulation-based algorithms, and mixed-integer programming (MIP/MILP) models as potential solutions. Snauwaert et al. [32] generate over 300 instances involving the similar principles used in Snauwaert and Vanhoucke [31]. Computation time forms the basis for comparisons since average and maximum CPU time as well as the percentage of instances not solved to optimality within a processing time of one hour have been employed as indicators of problem complexity. Kosztyán and Novák [33], on the other hand, explore the significance of project indicators in influencing project structures, complexity, duration, and resource demands and their impact on scheduling and resource allocation algorithms. The authors introduce the flexible structure generator (FSG) to transform conventional project database

instances into flexible instances, enabling the analysis of project indicators in a flexible project environment. The project indicators assess project complexity and demands, enabling analysis and classification of flexible projects. This study provides illustrative examples for calculating specific indicator values from matrix-based instances and running corresponding unit test suites for indicator types. The paper also discusses the role of minimal and maximal structure variants in bridging the gap between traditional and emerging flexible project scheduling algorithms.

With the increasing complexity of project networks, along with the intricate trade-offs between duration and cost, extensive research has been conducted into various aspects of scheduling and network generation. Research on project scheduling covers a wide range of subjects, from basic techniques such as CPM to the complexities of network generation and project instance evaluation. The studies discussed in this paper also underscore the importance of generating realistic and challenging problem instances, which are essential for testing and validating optimization algorithms across various project scheduling scenarios. Despite several studies, it is observed that research on network generators for project scheduling problems does not adequately address how the characteristics of the problems and their solution spaces can change under different arrangements of the project network. Furthermore, the practical impact of complexity on solution difficulties has not been sufficiently explored in the extant literature. This study, therefore, aims to address this gap by experimenting how network complexity influences the solution and processing requirements of one of the most intricate types of scheduling problems, the Pareto front in time-cost trade-off problem (TCTP).

This significant area in project scheduling research is commonly studied within the field, where each activity can be performed using various execution modes, each associated with different durations and costs. In practice, the normal duration of an activity corresponds to the resource level for which the duration cannot be shortened without increasing the direct cost of the activity. Nevertheless, in practical scenarios, tasks are often accelerated (or crashed) below their normal durations to avoid delay penalties or to reduce expenses incurred in the form of indirect costs [34]. Such accelerations come at a cost, creating a trade-off between time and expenses. While various versions of TCTP have been the focus of numerous studies, the most complex variant seeks to achieve the complete set of optimal solutions generated by the numerous combinations of execution modes. This complete set of optimal solutions is referred to as the Pareto front and contains mutually non-dominated solutions; that is, cost components cannot be improved without worsening the corresponding duration component, and vice versa.

The remainder of this paper is organized as follows. Section 2 describes the research methodology. In Section 3 the computational results outlined. Section 4 provides a detailed discussion of findings. Finally, concluding remarks on the present work are given in Section 5.

2. Research Methodology

In order to investigate the characteristics of the Pareto front time-cost trade-off problems and their solution spaces under different arrangements of the project network, also in order to experiment how topological structure of the underlying project networks influence the computational requirements, a new set of instances have been generated in this study. As discussed in Section 1, numerous random network generators have been proposed in the literature, including ProGen/Max [19], RanGen [21], DAGEN [20], and RanGen2 [22]. Among these, RanGen2 has been demonstrated to be superior to the aforesaid generators. RanGen2 has, therefore, been adopted for generation of the networks for the instances used in this study. RanGen2 incorporates several topological indicators based on predefined values for these indicators. These indicators include I1 (number of activities), I2 (serial/parallel indicator, SP), I3 (activity distribution indicator), I4 (short precedence relations indicator), I5 (long precedence relations indicator), and I6 (topological float of

activities) [22]. Of these indicators, I2 – also the main focus of current study – is particularly notable, as it is used to regulate the networks closeness to a truly serial or completely parallel graph. Section will cover the explanations about the instances generated.

The project test dataset consists of fictitious project networks with 25, 50, and 75 activities. Initially, project networks are generated using the activity-on-node (AoN) network generator RanGen2 [22]. These networks are designed with controlled variations in the SP-indicator (serial/parallel indicator, I2), which measures the extent to which a project network approximates a completely serial or parallel configuration [35]. The dataset includes networks where the SP-indicator ranges from 0.3 to 0.9, in steps of 0.3. Specifically, the values 0.3, 0.6, and 0.9 represent almost parallel, intermediate, and almost serial network configurations, respectively. Then, time-cost alternatives, ranging from 3 to 9 per activity, are generated using non-increasing convex time-cost functions, following the methodology outlined by Akkan et al. [36]. Finally, the new TCTPs are created by combining the RanGen2 networks with the execution modes generated through a custom C# code.

It is of utmost importance to note that for each problem size, the three problems (e.g., T25_3, T25_6, and T25_9) differ as follows. The three distinct problems are characterized by their network structure. Specifically, T25_3, T50_3, and T75_3 are generated using an I2 value of 0.3, T25_6, T50_6, and T75_6 using I2 of 0.6, and T25_9, T50_9, and T75_9 using I2 of 0.9. Despite these differences in network structure, for each problem size, all the three variants share identical execution modes per activity. This ensures that the observed variations are attributable solely to the differences in topological structure of networks, allowing for a focused analysis of its effects. Table 1 outlines the details of this dataset.

All the 25-activity instances consist of four activities with three modes, two with four modes, a single one with five modes, four with six modes, four with seven modes, six with eight modes, and four with nine modes; totaling to 3.47×10^{19} different time-cost combinations. T50 problems consist of nine activities with three modes, seven with four modes, seven with five modes, three with six modes, nine with seven modes, nine with eight modes, and six with nine modes; totaling to 1.57×10^{37} different possible combinations. And T75_3 to T75_9 problems consist of seven activities with three modes, 16 with four modes, 18 with five modes, nine with six modes, six with seven modes, ten with eight modes, and nine with nine modes; totaling to 1.77×10^{55} different time-cost realizations.

Table 1. Details of T25, T50, and T75 problems

Problem	# of Acts.	# of Modes	I2 (Nominal)*
T25_3	25	[3-9]	0.30
T25_6	25	[3-9]	0.60
T25_9	25	[3-9]	0.90
T50_3	50	[3-9]	0.30
T50_6	50	[3-9]	0.60
T50_9	50	[3-9]	0.90
T75_3	75	[3-9]	0.30
T75_6	75	[3-9]	0.60
T75_9	75	[3-9]	0.90

*The nominal value entered into RanGen2 may differ from the actual value observed.

2.1. Case Example

In this subsection, a detailed case example is presented to illustrate the structure and characteristics of the dataset used in the subsequent stages of this study. As shown in Table 2, this example includes five actual and two milestone activities with a solution space comprising 540 possible realizations. The purpose of this example is to elucidate the rationale behind the generation of the problems, specifically focusing on the variations in network structures and their implications. The same rationale employed for generating the T25, T50, and T75 problems has been applied to create the T5_3, T5_6, and T5_9 case problems. Each set of problems is designed with varying network structures according to their respective I2 values. Specifically, T5_3, T5_6, and T5_9, like their T25, T50, and T75 counterparts, are distinguished by I2 values of 0.3, 0.6, and 0.9, respectively. It is worth noting that Table 2 provides details on the mutual execution modes for the three T5 problems. Moreover, the contrast in precedence relationships among the T5 case examples is detailed in Table 3.

In addition, the structural differences among the different versions of T5 problem is showcased by presenting their respective AoN diagrams in Figs. 1 to 3. These graphical representations clarify how each problem is configured and how the networks differ in terms of their I2 values, further, they provide insights into how the variations in network topology contribute to the overall problem design.

Each project network for the T5 case examples was analyzed using a custom graph analyzer coded in C#. This tool calculates several key metrics, namely, the total number of paths, the longest arc in the network, and the actual SP-indicator (I2) value. It is crucial to measure the actual I2 value since, during the network generation process with RanGen2, the I2 values are initially specified as network generation parameters.

Table 2. Mutual execution modes for T5 case examples

	Mode 1		Mode 2		Mode 3		Mode 4		Mode 5	
	Dur. (day)	Cost ($)	Dur. (day)	Cost ($)	Dur. (day)	Cost ($)	Dur. (day)	Cost ($)	Dur. (day)	Cost ($)
Start	0	0	-	-	-	-	-	-	-	-
1	5	87,204	16	80,095	36	70,476	45	67,635	-	-
2	1	84,840	32	63,943	39	62,595	-	-	-	-
3	6	39,755	27	34,692	52	30,576	-	-	-	-
4	6	92,243	14	86,072	25	79,323	36	74,888	42	73,626
5	9	50,307	23	45,166	42	42,840	-	-	-	-
Finish	0	0	-	-	-	-	-	-	-	-

Table 3. Precedence relationships for T5 case examples

Act.	Direct Predecessor(s)		
	T5_3	*T5_6*	*T5_9*
Start	-	-	-
1	Start	Start	Start
2	Start	1	1
3	1, 2	Start	1
4	1	2, 3	2, 3
5	1	2, 3	4
Finish	3, 4, 5	4, 5	5

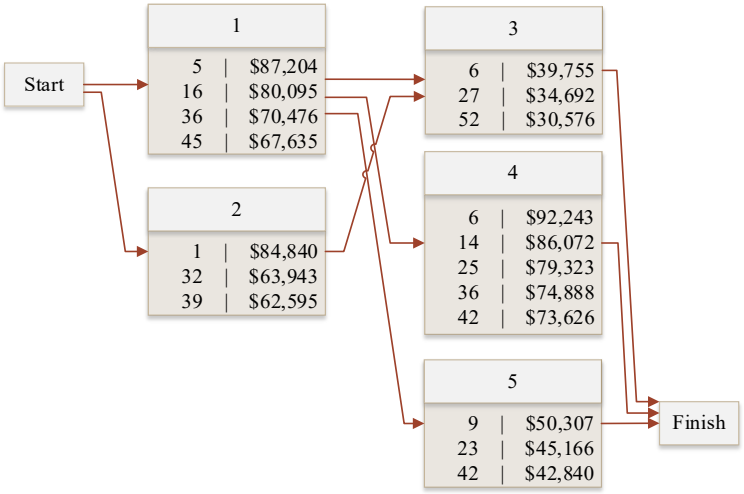

Fig. 1. Network diagram for T5_3 case example

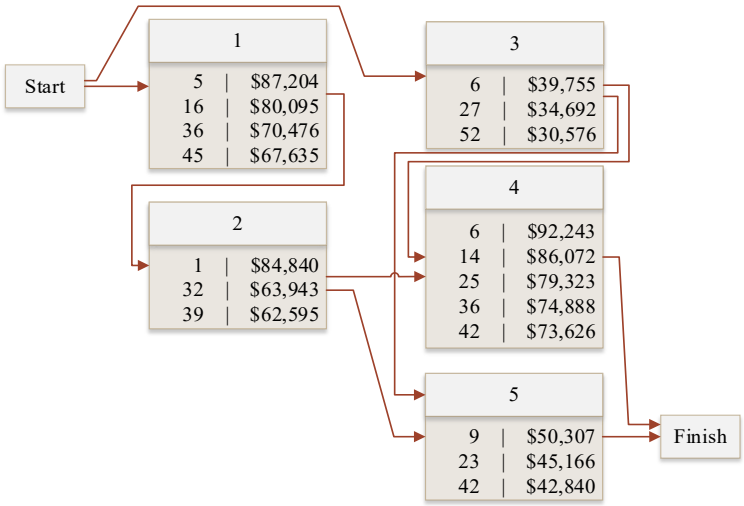

Fig. 2. Network diagram for T5_6 case example

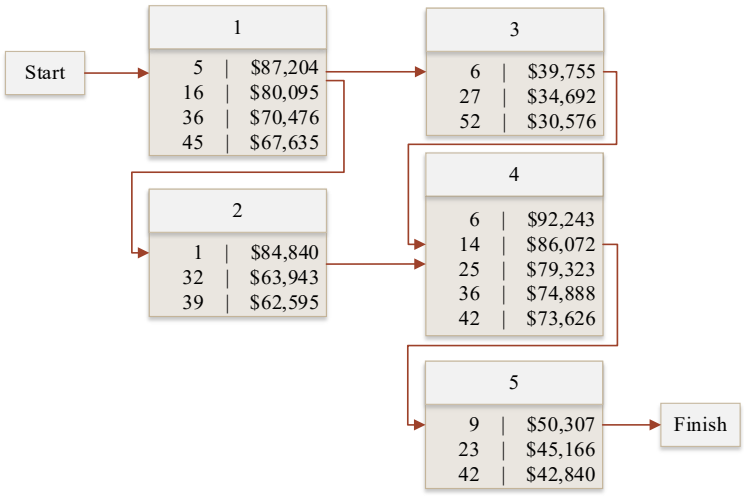

Fig. 3. Network diagram for T5_9 case example

However, these nominal values are experienced to differ slightly from the actual values observed in the generated networks. The post-analysis conducted using the custom tool reveals these slight discrepancies, providing a more accurate measurement of the I2 value as provided in Table 4. This small variation arises due to the fact that the nominal (pre-specified) I2 values either could not be generated or simply do not exist.

3. Computational Results

As mentioned earlier, this study's objective is to explore the practical impact of network complexity on solution difficulty by examining how varying topological structure of the network affects the computational demand of solving one of the most intricate scheduling problems, the Pareto front time-cost trade-off problem (TCTP). Vanhoucke and Maenhout [24] emphasize that the hardness of a problem instance is typically measured by the CPU time required for an exact solution procedure to solve it to optimality. In line with this perspective, Pareto front optimization of the new set of instances discussed earlier was conducted using a basic mixed-integer linear programming (MILP) model that uses the Gurobi Optimizer (version 11.0). The specifics of the MILP model are not elaborated upon here, as the primary focus of this study is not on the model's development or its intricacies. Instead, the emphasis is on employing a uniform technique to evaluate how different problem characteristics impact the performance of the MILP approach. The optimization routines are implemented using C# on a 64-bit platform. The experiments were conducted on a desktop computer equipped with 8 GB of RAM, an Intel® Core™ i7-9700 CPU @ 3.00 GHz, and a 64-bit Windows 11 operating system.

Initially, each project network for the T25, T50, and T75 problems was analyzed using a custom graph analyzer developed in C#. This tool, introduced earlier in the context of the T5 case example, was again employed to calculate key network characteristics, including the total number of paths, the length of the longest arc, and the actual I2 values as given in Table 5. As discussed previously, while RanGen2 allows for the input of desired I2 values during network generation, these values may differ slightly from the actual I2 observed post-generation. The minor discrepancies between the nominal and actual I2 values stems from the technique RanGen2 uses to approximate the predefined I2 value. This approximation is influenced by I1 (number of activities) indicator and the length of the longest arc, which RanGen2 balances to create a network structure as close as possible to the pre-specified I2 value.

Table 4. Topological characteristics of T5 case examples

Problem	Total # of Paths	Longest Arc	I2 (Nominal)	I2 (Actual)
T5_3	4	2	0.30	0.25
T5_6	4	3	0.60	0.50
T5_9	2	4	0.90	0.75

Table 5. Topological characteristics of T25, T50, and T75 problems

Problem	Total # of Paths	Longest Arc	I2 (Actual)
T25_3	510	8	0.29
T25_6	168	15	0.58
T25_9	8	22	0.88
T50_3	9,998	15	0.29
T50_6	72,000	30	0.59
T50_9	32	45	0.90
T75_3	7,944,022	23	0.30
T75_6	2,663,424	45	0.59
T75_9	256	67	0.89

As can be followed from Table 5, as I2 increases, the longest arc in the problem also increases, confirming that higher values of I2 correspond to networks where activities are arranged in a more serial structure. Further, there is a noticeable decrease in the total number of paths as I2 increases, reinforcing that higher values of I2 result in greater number of parallel paths.

3.1. T5 case examples

The Pareto front optimization is first applied to the T5 case examples. The Pareto fronts obtained for T5_3, T5_6, and T5_9 are plotted against each other in Fig. 4. As seen in Fig. 4, as I2 increases from 0.3 for T5_3 to 0.9 for T5_9, the number of non-dominated solutions increases. This indicates that more diverse solutions become available as the project network approaches a serial structure. The spread of the Pareto front becomes wider with higher values of I2. T5_9, with the highest I2 of 0.9, shows a more extended range of solutions, reflecting greater variability in the solution space. More precisely, hyperarea which measures the diversity (distribution and spread) of the Pareto fronts is calculated as 10,371,404, 9,135,274, and 7,564,477 for T5_3, T5_6, and T5_9 case examples, respectively. In addition, the Pareto front corresponding to the smallest I2 is noticeably steeper. This steepness implies that for parallel/pseudo-parallel networks, small increases in duration result in significant reductions in cost.

3.2. T25 problems

Following the analysis of the T5 case example, this section investigates the T25 problems. The results obtained by the MILP model for the T25 problems are presented in Table 6. This table presents CPU time, the normal (cheapest and longest) solution, the crashed (most expensive and shortest) solution, as well as the number of Pareto front (PF) solutions, per each setting. The costs considered are solely the direct costs of the activities, where in the normal solution, all activities are executed in their normal modes, and in the crashed solution, all activities are executed in their crashed modes. It must be pointed out that crashed solutions are not derived through an optimization process; rather, all activities are crashed to the maximum extent possible. As a result, some fully crashed activities may have float.

As seen in Table 6, I2 indicator's impact on the results can be observed through variations in the performance metrics. Although all variants of the T25 problem have the same total time-cost mode combination of 3.47×10^{19}, varying I2 has dramatically altered the solution space. When CPU time is considered, for I2 = 0.3 it is the highest at 49.2 seconds, while for I2 = 0.9, it is the lowest at 21.13 seconds.

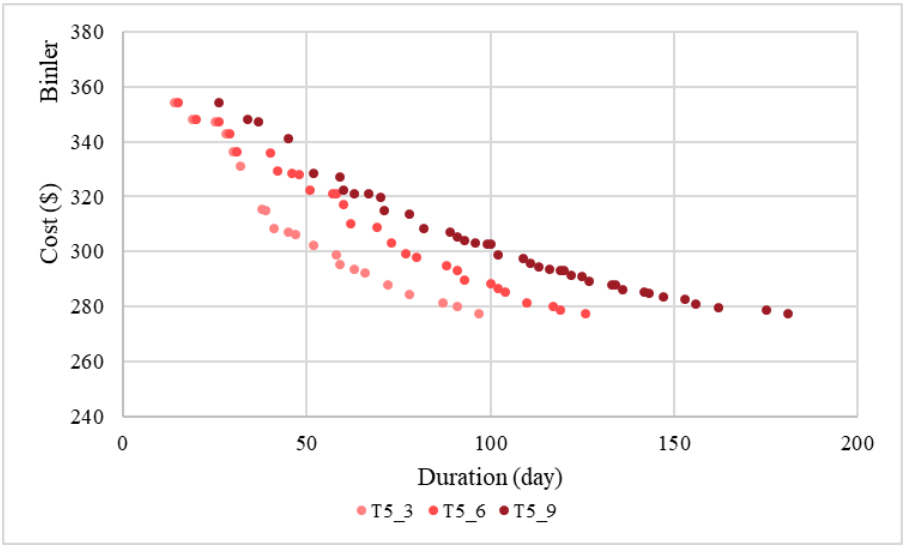

Fig. 4. Pareto fronts for T5 problems

Table 6. Comparison of results for T25 problems

Problem	CPU Time (S)	Normal Sol.		Crashed Sol.		# of PF Sols
		Dur. (day)	Cost ($)	Dur. (day)	Cost ($)	
T25_3	49.20	388	1,571,636	41	1,986,464	311
T25_6	36.36	720	1,571,636	64	1,999,585	598
T25_9	21.13	1,033	1,571,636	91	2,003,340	902

With regard to the number of Pareto front solutions, for I2 = 0.3, it is 311, and this number rises to 902 for I2 = 0.9. Despite T25_9 having a larger set of Pareto front solutions, it requires less CPU time compared to T25_3 and T25_6. Similarly, T25_6 also shows a shorter CPU time compared to T25_3, in spite of having a greater number of non-dominated solutions in its front. There is an increase in CPU time from I2 = 0.9 to 0.6 by 72.04%, while from I2 = 0.6 to 0.3 the increase is 35.31%. In the same order, there is a decline in the size of Pareto front set by 47.99% and 33.70%, respectively. This pattern highlights an intriguing aspect of the problem's computational complexity. This trade-off between CPU time and the number of Pareto front solutions is demonstrated in Fig. 5 for each setting of T25 problem.

Fig. 6 displays the Pareto fronts for T25_3, T25_6, and T25_9 problems plotted against one another. As seen in this figure, the number of non-dominated solutions grows as I2 rises from 0.3 for T25_3 to 0.9 for T25_9. In addition, Hyperarea for the Pareto fronts is calculated as 381,831,632, 335,546,722, and 268,809,617 for T25_3, T25_6, and T25_9 problems, respectively. This suggests that when the project network becomes closer to a serial structure, Pareto front stretches out, leading to a wider range of non-dominated solutions. Moreover, the time-cost curve associated with the smallest I2 has a much sharper slope. This steepness suggests that tiny duration increases lead to considerable cost savings for parallel and pseudo-parallel networks. A steeper Pareto front curve, on the other hand, can increase the complexity of the problem, as it poses a more challenging solution space for exploration compared to a shallower curve.

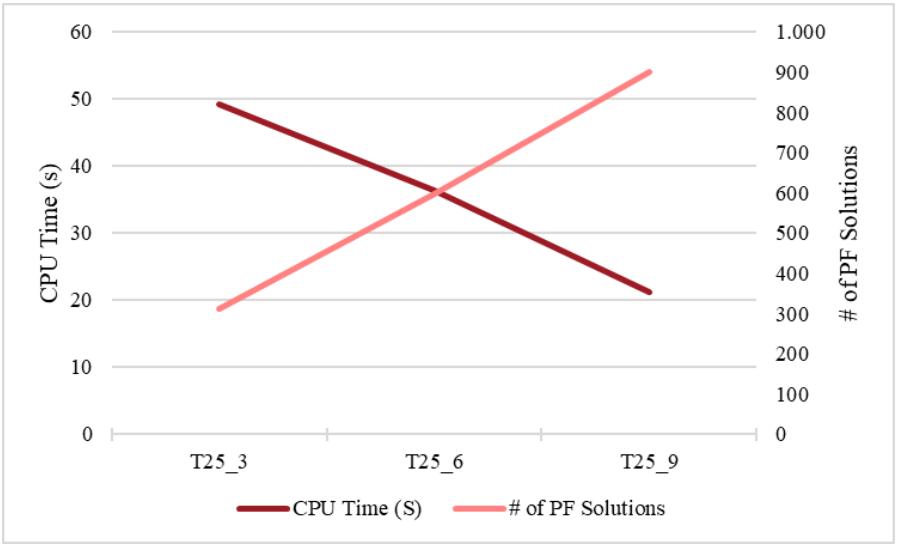

Fig. 5. CPU Time vs. no. of distinct Pareto front solutions for T25 problems

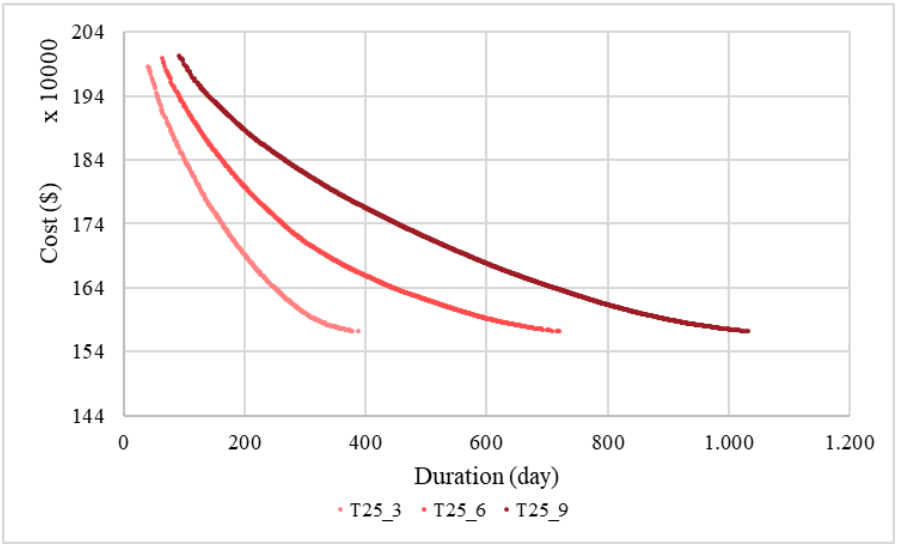

Fig. 6. Pareto fronts for T25 problems

3.3. T50 problems

This section explores the T50 problems, providing a detailed analysis of their characteristics and the results obtained. The results of the MILP model for the T50 problems are presented in Table 7 which shows CPU time, the normal (cheapest and longest) solution, the crashed (most expensive and shortest) solution, and the number of Pareto front (PF) solutions, for each variant of the problem. Similar to T25 set, the costs considered are solely the direct costs of the activities, and that crashed solutions are not derived through an optimization process. They obtained by crashing all activities are to the maximum extent possible, therefore, some crashed activities may have float.

As seen in Table 7, the impact of I2 on the performance metrics is present here as well. Despite all variants of the T50 problem share the same total time-cost realizations of 1.57×10^{37}, varying I2 has significantly transformed the solution space. With regard to CPU time, the pattern is similar to the previous dataset. CPU time decreases as I2 increases with 279.86 seconds for I2 = 0.3, 178.85 seconds for I2 = 0.6, and 79.47 seconds for I2 = 0.9. When the number of Pareto front solutions is considered, the observations are consistent with the previous dataset and shows that higher values of I2 lead to a larger set of optimal solutions. More specifically, for I2 = 0.3, it is 577, and this number rises to 1,835 for I2 = 0.9. Similar to the observations made for the T25 problems, a notable trade-off between the number of Pareto front solutions and CPU time is also evident for the T50 problems. As previously discussed, although it is expected that a higher number of Pareto front solutions would require increased CPU time, T50_9 with a larger set of Pareto front solutions requires less processing time compared to T50_3 or T50_6.

Table 7. Comparison of results for T50 problems

Problem	CPU Time (S)	Normal Sol.		Crashed Sol.		# of PF Sols
		Dur. (day)	Cost ($)	Dur. (day)	Cost ($)	
T50_3	279.86	699	2,931,049	77	3,627,997	577
T50_6	178.85	1,407	2,931,049	165	3,708,208	1,219
T50_9	79.47	2,074	2,931,049	221	3,718,023	1,835

Similarly, T50_6 also requires a shorter computational time compared to T50_3, despite having a greater number of non-dominated solutions. There is an increase in CPU time from I2 = 0.9 to 0.6 by 125.06%, while from I2 = 0.6 to 0.3 the increase is 56.47%. In the same order, there is a decline in the size of Pareto front set by 52.67% and 33.57%, respectively. This trade-off between CPU time and the number of Pareto front solutions for T50 problems is displayed in Fig. 7.

Fig. 8 illustrates the Pareto fronts for the T50_3, T50_6, and T50_9 problems plotted together. The figure reveals that as the project network approaches a more serial structure, the Pareto front expands, resulting in a broader range of non-dominated solutions. More specifically, Hyperarea values associated with these fronts are calculated as 1,453,752,001, 1,212,075,679, and 958,423,761 for T50_3, T50_6, and T50_9 problems, respectively. These observations are consistent with the previous findings for the T25 problems. Additionally, the time-cost curve for the lowest I2 value, again, exhibits a much steeper slope.

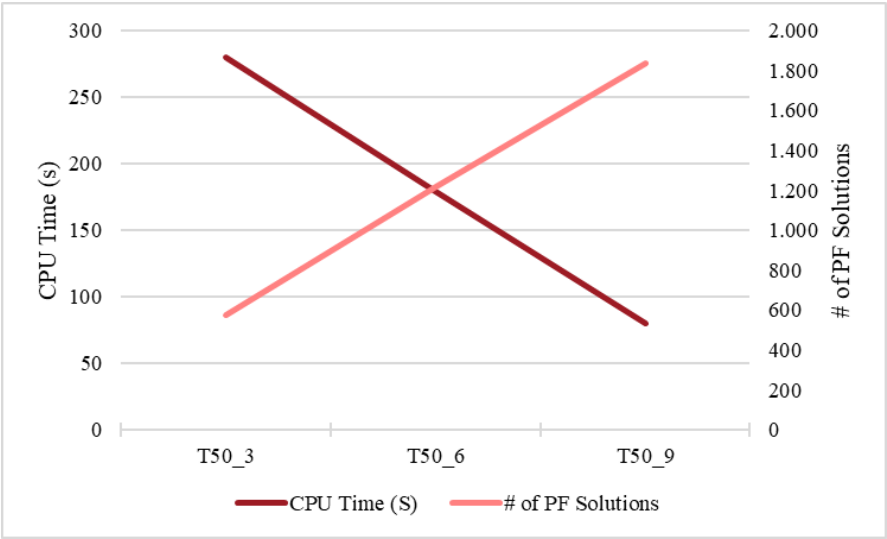

Fig. 7. CPU time vs. no. of distinct Pareto front solutions for T50 problems

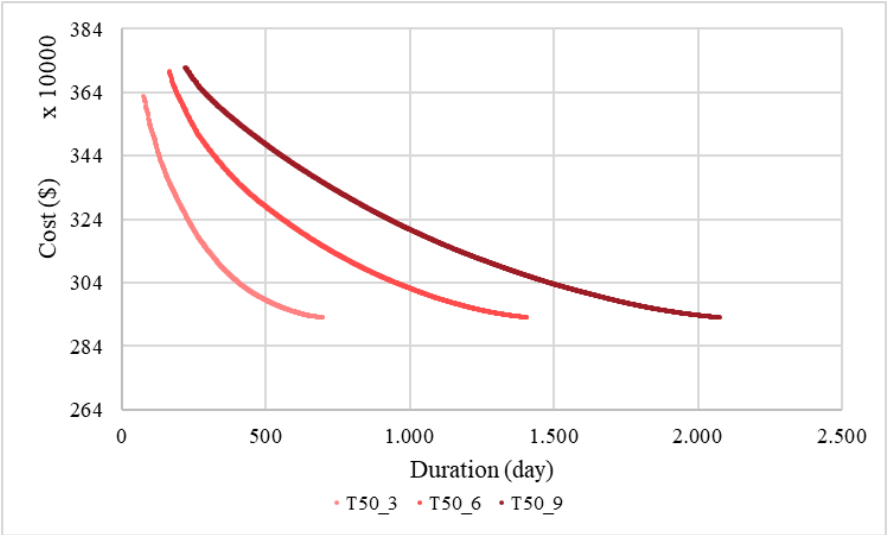

Fig. 8. Pareto fronts for T50 problems

This steepness implies that smaller values of I2 can heighten the problem's complexity, making the solution space more challenging to explore compared to a shallower curve.

3.4. T75 problems

This section further sheds light on the characteristics of T75_3, T75_6 and T75_9, presenting the solutions achieved for these problems. The results of the MILP model for the T75 problems are presented in Table 8 which includes CPU time, the normal and crashed solutions, and the number of Pareto front (PF) solutions, for each version of the problem. Crashed solutions are obtained in a manner similar to T25 and T50 problems. That is, solely the direct costs are considered, and that crashed solutions are obtained by crashing all activities are to the maximum extent possible, which means some crashed activities may have float.

As provided in Table 8, the influence of the I2 indicator on the performance metrics is evident here as well. Even though all T75 problems have the identical execution modes, totaling to 1.77×10^{55} number of different combinations, changes in I2 have remarkably reshaped the solution space of the three variants. Regarding CPU time, the pattern observed is consistent with that of T25 and T50 problems. Increasing I2 leads to significant reductions in CPU time with 16,837.01 seconds for I2 = 0.3, 662.43 seconds for I2 = 0.6, and 129.47 seconds for I2 = 0.9. The extraordinarily long CPU time for T75_3, despite having the lowest number of Pareto front solutions of 941 and the smallest longest arc of 23, can be attributed to the exceptionally high number of parallel paths, which totals 7,944,022. This number of parallel paths is 198% higher than that for T75_6 and 3.1×10^{6}% higher than for T75_9 problem. Considering the number of Pareto front solutions, the observations are in line with those from previous datasets and demonstrate that higher values of I2 lead to a larger set of Pareto solutions. More precisely, 941 for I2 = 0.3, 1,873 for I2 = 0.6, and 2,802 for I2 = 0.9. Consistent with the observations for the T25 and T50 problems, a significant trade-off between the number of Pareto front solutions and CPU time is also evident for the T75 problems. As previously noted, while locating a higher number of Pareto front solutions would generally demands more CPU time, T75_9, which has a larger set of Pareto front solutions, requires less processing time compared to T75_3 and T75_6. Likewise, T75_6, despite having more non-dominated solutions than T75_3, also demands shorter computational time. A notable increase in CPU time is observed from I2 = 0.9 to 0.6 by 2,441.72%, while from I2 = 0.6 to 0.3 the increase is 411.66%. In the same order, there is a decline in the size of Pareto front set by 49.76% and 33.15%, respectively. Fig. 9 clearly demonstrates the trade-off between CPU time and the number of Pareto front solutions for the T75 problems.

Pareto fronts for T75_3, T75_6, and T75_9 problems are shown in Fig. 10. This figure reveals that as the project network lies closer to a serial structure, the Pareto front broadens, leading to larger range of pareto solutions. To be precise, Hyperarea values associated with these Pareto fronts are calculated as 3,570,038,770, 3,058,874,865, and 2,420,847,312 for T75_3, T75_6, and T75_9 problems, respectively. These findings align with the observations made for the T25 and T50 problems. Moreover, the efficient frontier for the smallest level of I2 exhibits a significantly steeper slope.

Table 8. Comparison of results for T75 problems

Problem	CPU Time (S)	Normal Sol		Crashed Sol		# of PF Sols
		Dur. (day)	Cost ($)	Dur. (day)	Cost ($)	
T75_3	16,837.01	1,103	4,678,002	127	5,911,510	941
T75_6	662.43	2,103	4,678,002	216	5,944,992	1,873
T75_9	129.47	3,127	4,678,002	295	5,976,856	2,802

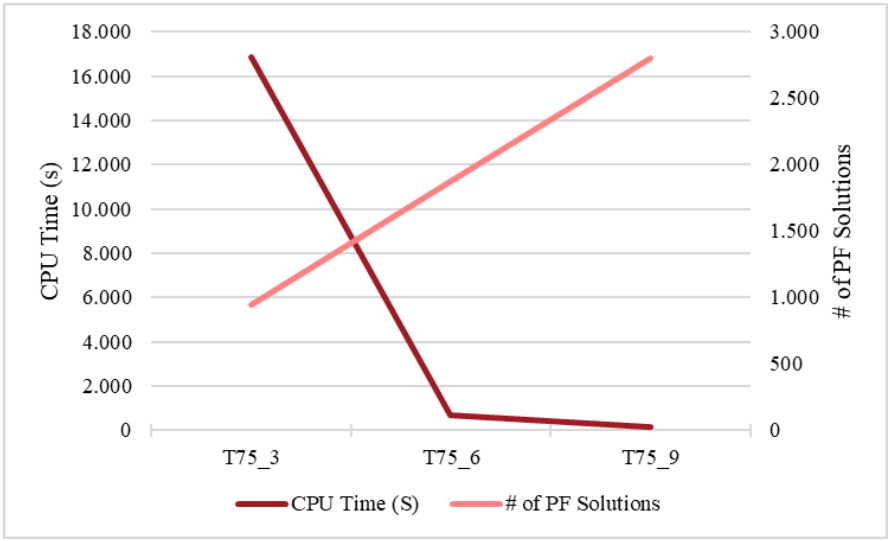

Fig. 9. CPU time vs. no. of distinct Pareto front solutions for T75 problems

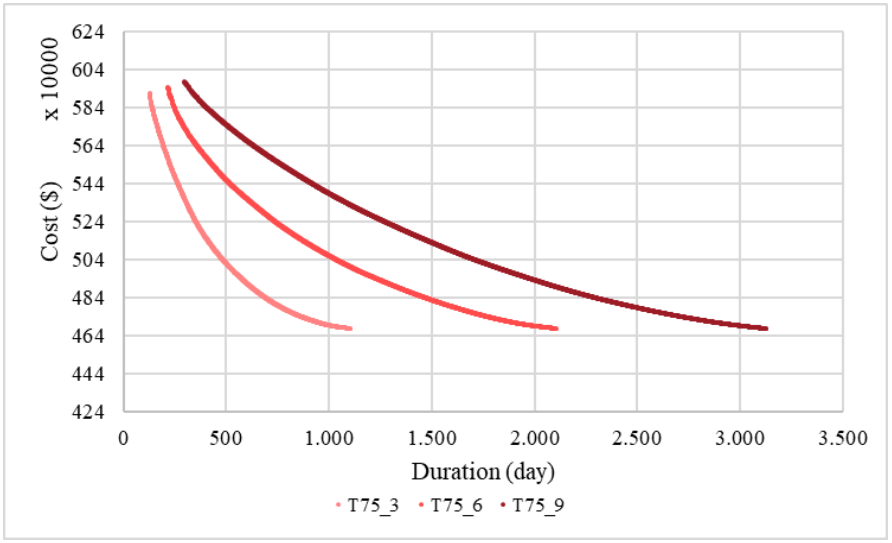

Fig. 10. Pareto fronts for T75 problems

This steepness suggests that larger values of I2 can reduce the complexity of the problem, making the solution space easier to explore.

4. Discussion of Findings

The performance evaluations across the different problem instances reveal that the spread of the Pareto front becomes wider with higher values of I2, indicating a broader range of non-dominated solutions as the project network transitions towards a more serial structure. A clear relationship between network topology and computational complexity is also observed, with the empirical hardness of the instances decreasing for increasing values of I2, leading to improved performance of the optimization algorithm. Empirical complexity, meanwhile, refers to the observed difficulty in solving the problems, reflected by the computational effort required for the solution of the instances. Interestingly, an inverse relationship between CPU time and the number of Pareto front solutions was observed, contrasting with the direct relationship between CPU time and the number of

paths in the network. Contrary to the expectation that locating a higher number of Pareto front solutions would generally demand more CPU time, the experiments show that smaller I2 values with fewer Pareto solution demand much higher computational effort. Specifically, lower I2 values, which result in a greater number of paths and a steeper Pareto front, are significantly more challenging and require higher computational resources. This can be explained by the observation that the efficient front curve is steeper for smaller I2 values, making the solution space more difficult for algorithms to explore effectively.

A critical insight from this study challenges the common assumption that empirical hardness of a problem is best reflected by the total number of time-cost realizations. This notion is debunked here, as the experiments with problems having identical numbers of realizations but varying I2 values demonstrate that the number of realizations alone may not accurately portray the true complexity of the problems. Instead, the structural characteristics of the network, influenced by I2, play a crucial role in determining the problem's computational demands. The increased computational effort primarily arises from the way the non-dominated solutions are distributed in the solution space. Problems with steeper Pareto front curves make it more challenging for algorithms to converge to optimal solutions because a small change in one objective can cause substantial changes in the value of another objective. It can be said that, in problems characterized by steeper curves, the trade-off between project time and cost is more pronounced and noticeable. In contrast, a shallower Pareto front curve allows for easier and less computationally expensive exploration. Consequently, locating a larger number of Pareto solutions on a shallow curve can be significantly less time-consuming compared to finding fewer solutions on a steeper curve. In light of these observations, this study offers valuable insights and guidelines on how network topology influences computational complexity, highlighting the importance of considering I2 in problem-solving strategies.

5. Conclusions

This study experimented on how network complexity influences the solution and processing requirements of one of the most intricate types of scheduling problems, the Pareto front in time-cost trade-off problem (TCTP). In order to investigate the characteristics of the Pareto front time-cost trade-off problems and their solution spaces under different arrangements of the project network, and to experiment with how the topological structure of the underlying project networks influences the computational requirements, a new set of instances was generated in this study. The results from the computational experiments reveal several key insights. Firstly, the performance of the proposed procedures improved for instances with larger serial/parallel indicator (SP or I2) values, which are closer to serial networks. This indicates that the optimization algorithms are more effective when dealing with networks that lie closer to a serial graph.

The experiments conducted at three activity levels demonstrated that the percentage of optimal solutions for the mixed-integer linear programming (MILP) model increases with higher I2 values. This further emphasizes the significant impact of the I2 indicator on the problem's solution space. Contrary to the common expectation that a higher number of Pareto front solutions would generally demand more CPU time, the findings show that smaller I2 values, which correspond to steeper time-cost curves, require significantly higher computational effort. For instance, a solution space with fewer Pareto front solutions may still require considerable computational resources due to its topological structure. This suggests that the steepness of the Pareto front curve, rather than the number of Pareto front solutions alone, plays a crucial role in determining the computational complexity of the problem. Additionally, the study found that project complexity does not necessarily correlate with the total number of time-cost realizations. Instead, the empirical hardness of the instances increases with decreasing I2 values. This means that the complexity of the problem is more accurately reflected by the network's topological structure

than by the sheer number of time-cost combinations.

To sum up, the number of time-cost combinations or the number of Pareto front solutions that must be explored and located per se may not truly portray the complexity of the problems. Drawing analogies with similar problems in other contexts may provide additional insights into the issues raised by this study. Furthermore, future research could focus on developing adaptive algorithms that adjust their strategies based on network topology, potentially improving efficiency in solving TCTP problems. Future research focusing on empirically validating the performance trends observed with MILP by applying both heuristic and metaheuristic methods to various network structures and complexities seems to be another promising avenue.

Declaration

Funding

This research received no external funding.

Author Contributions

S. Aminbakhsh: Conceptualization, Methodology, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing.

Acknowledgments

Not applicable.

Data Availability Statement

The data presented in this study are available on request from the corresponding author.

Ethics Committee Permission

Not applicable.

Conflict of Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

[1] Kelley JE, Walker MR (1959) Critical-path planning and scheduling. In: Proceedings of Eastern Joint Computer Conference 16:160-173. https://doi.org/10.1145/1460299.1460318

[2] Hegazy T (2001) Critical path method–line of balance model for efficient scheduling of repetitive construction projects. Transportation Research Record 1761(1):124-129. https://doi.org/10.3141/1761-16

[3] Ökmen Ö (2013) A procedure for critical path method-based scheduling in linear construction projects. Journal of The South African Institution of Civil Engineering 55(2):12-20.

[4] Shankar NR, Rao PP, Siresha S, Madhuri KU (2011) Critical path method in a project network using ant colony optimization. International Journal of Computational Intelligence Research 7(1):7-16.

[5] Altun M, Sonmez R, Akcamete A (2020) A mixed integer programming method for multi-project resource leveling. Journal of Construction Engineering, Management & Innovation 3(2):131-140. https://doi.org/10.31462/jcemi.2020.02131140

[6] Bettemir ÖH, Erzurum T (2019) Comparison of resource distribution metrics on multi-resource projects. Journal of Construction Engineering Management & Innovation 2(2):93-102. https://doi.org/10.31462/jcemi.2019.02093102

[7] Bettemir ÖH, Birgönül MT (2016) Network analysis algorithm for the solution of discrete time-cost trade-off problem. KSCE Journal of Civil Engineering 21(4):1047-1058. https://doi.org/10.1007/s12205-016-1615-x

[8] Bettemir ÖH, Erzurum T (2021) Exact solution of resource leveling problem by exhaustive enumeration with parallel programming. Teknik Dergi 32(3):10767-10805. https://doi.org/10.18400/tekderg.595238

[9] Pascoe TL (1966) Allocation of resources C.P.M. Revue Francaise Recherche Operationelle 10(38):31-38.

[10] Davies EM (1973) An experimental investigation of resource allocation in multiactivity projects. Journal of the Operational Research Society 24(4):587-591. https://doi.org/10.2307/3008335

[11] Kaimann RA (1974) Coefficient of network complexity. Management Science 21(2):172-77.

[12] Davis EW (1975) Project network summary measures constrained- resource scheduling. AIIE Transactions 7(2):132-142. https://doi.org/10.1080/05695557508974995

[13] Mastor AA (1970) An experimental investigation and comparative evaluation of production line balancing techniques. Management Science 16(11):728-746. https://doi.org/10.1287/mnsc.16.11.728

[14] Elmaghraby SE, Herroelen W (1980) On the measurement of complexity in activity networks. European Journal of Operational Research 5(4):223-234. https://doi.org/10.1016/0377-2217(80)90053-3

[15] Herroelen WS, Caestecker G (1979) The generation of random activity networks, Research report No. 7906, Department of Applied Economics, K.U. Leuven. https://lirias.kuleuven.be/1833996&lang=en. Accessed 01 Aug 2024.

[16] Liu CL, Chen HY (1991) Critical path for an interval project network. Journal of Management Sciences in China 9(1):27-32.

[17] Bein WW, Kamburowski J, Stallmann MFM (1992) Optimal reduction of Two-Terminal directed acyclic graphs. SIAM Journal on Computing 21(6):1112-1129. https://doi.org/10.1137/0221065

[18] Kolisch R, Sprecher A, Drexl A (1995) Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 41:1693-1703.

[19] Schwindt C (1995) ProGen/Max: A New Problem Generator for Different Resource-Constrained Project Scheduling Problems with Minimal and Maximal Time Lags. Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe, WIOR Report 449.

[20] Agrawal M, Elmaghraby S, Herroelen W (1996) DAGEN: A generator of testsets for project activity nets. European Journal of Operational Research 90(2):376-382. https://doi.org/10.1016/0377-2217(95)00361-4

[21] Demeulemeester E, Vanhoucke M, Herroelen W (2003) RanGen: A random network generator for activity-on-the-node networks. Journal of Scheduling 6:17-38. https://doi.org/10.1023/A:1022283403119

[22] Vanhoucke M, Coelho J, Debels D, Maenhout B, Tavares LV (2008) An evaluation of the adequacy of project network generators with systematically sampled networks. European Journal of Operational Research 187(2):511-524. https://doi.org/10.1016/j.ejor.2007.03.032

[23] Tavares LV (1999) Advanced Models for Project Management, Sixteenth Edition. Springer Science & Business Media.

[24] Vanhoucke M, Maenhout B (2009) On the characterization and generation of nurse scheduling problem instances. European Journal of Operational Research 196(2):457-467. https://doi.org/10.1016/j.ejor.2008.03.044

[25] Batselier J, Vanhoucke M (2017) Project regularity: Development and evaluation of a new project characteristic. Journal of Systems Science and Systems Engineering 26(1):100-120. https://doi.org/10.1007/s11518-016-5312-6

[26] Ellinas C, Allan N, Johansson A (2018) Toward project complexity evaluation: A Structural perspective. IEEE Systems Journal 12(1):228-239. https://doi.org/10.1109/jsyst.2016.2562358

[27] Van Den Eeckhout M, Maenhout B, Vanhoucke M (2020) Mode generation rules to define activity flexibility for the integrated project staffing problem with discrete time/resource trade-offs. Annals of Operations Research 292(1):133-160. https://doi.org/10.1007/s10479-020-03619-3

[28] Thiele B, Ryan MJ, Abbasi A (2021) Developing a dataset of real projects for portfolio, program and project control management research. Data in Brief 34:106659. https://doi.org/10.1016/j.dib.2020.106659

[29] Queiroz M, Lucas F, Sörensen K (2022) Instance generation tool for on-demand transportation problems. European Journal of Operational Research 317(3):696-717.

[30] Coelho J, Vanhoucke M (2023) New resource-constrained project scheduling instances for testing (meta-)heuristic scheduling algorithms. Computers & Operations Research 153:106165. https://doi.org/10.1016/j.cor.2023.106165

[31] Snauwaert J, Vanhoucke M (2023) A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem. European Journal of Operational Research 307(1):1-19. https://doi.org/10.1016/j.ejor.2022.05.049

[32] Snauwaert J, Van Eynde R, Vanhoucke M (2023) On the complexity of efficient multi-skilled team

composition. Computers & Operations Research 157:106277. https://doi.org/10.1016/j.cor.2023.106277

[33] Kosztyán ZT, Novák G (2024) Project indicators and flexible project structure generators. Journal of Computational Science 75:102203. https://doi.org/10.1016/j.jocs.2023.102203

[34] Sonmez R, Aminbakhsh S, Atan T (2020) Activity uncrashing heuristic with noncritical activity rescheduling method for the discrete time-cost trade-off problem. Journal of Construction Engineering and Management 146(8). https://doi.org/10.1061/(asce)co.1943-7862.0001870

[35] Vanhoucke M (2010) Using activity sensitivity and network topology information to monitor project time performance. Omega 38(5):359-370. https://doi.org/10.1016/j.omega.2009.10.001

[36] Akkan C, Drexl A, Kimms A (2005) Network decomposition-based benchmark results for the discrete time–cost tradeoff problem. European Journal of Operational Research 165(2):339-358. https://doi.org/10.1016/j.ejor.2004.04.006