applied sciences

MDPI

Article

Genetic Programming for Dynamic Resource Allocation in Construction Projects: A Simulation-Based Case Study

Barbara Strug [1,*], Joanna Strug [2] and Adam Górski [1]

[1] Institute of Applied Computer Science, Jagiellonian University, Lojasiewicza 11, 30-348 Krakow, Poland; a.gorski@uj.edu.pl
[2] Faculty of Electrical and Computer Engineering, PK Krakow University of Technology, ul. Warszawska 24, 31-155 Krakow, Poland; joanna.strug@pk.edu.pl
* Correspondence: barbara.strug@uj.edu.pl

Featured Application

Improving resource allocation at the building/construction site, helping to reduce construction time and costs, as well as improving the usage of available resources by limiting the stand-by time.

Abstract

Construction projects require continuous allocation of heterogeneous resources such as workers, teams, and machinery. Traditional scheduling and assignment methods often assume static conditions and try to adapt when site conditions change or delays occur. This paper proposes a Genetic Programming (GP)–based framework to evolve resource allocation rules that adapt dynamically during project execution. Each GP individual represents a decision rule that maps the current construction-site state to resource–task assignments. The evolved rules are evaluated through simulation on a case study of a residential building and compared with the random resource allocation baseline approach. Results show that GP-generated rules can reduce project makespan, and improve resource utilization, as the experiments have shown an average cost of 1157.57 vs. 3085 for the baseline, which is about 62.5% lower, with a best of 813 vs. 2025 over 30 runs The findings highlight the potential of GP as a tool for dynamic resource allocation in construction management.

Keywords: BIM; task graph; genetic programming; task allocation; construction site optimization; scheduling

Check for updates

Academic Editor: Andrea Prati

Received: 14 October 2026
Revised: 28 October 2026
Accepted: 29 October 2026
Published: 30 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The construction industry accounts for around 13% of global Gross Domestic Product (GDP) [1,2], with an even higher share in developing countries, and employs around 7% of the global workforce. Together with its supply chain and other surrounding industries, its importance to global and local economies is much higher. As such, any delays in the construction project impact many participants. Main causes of delays include poor project planning and inadequate scheduling.

Efficient scheduling and resource allocation are central to successful construction project management. Workers, teams, and equipment must be assigned to tasks with different skill requirements, time, and dependencies. The challenge is higher than in many other domains because of the dynamic nature of construction environments, where delays, equipment problems, and changing site conditions, like weather issues, require continuous re-calculation of resource assignments [3,4].

Most classical approaches, such as integer programming, heuristic scheduling, or rule-based methods, are designed for static or semi-static conditions. When problems occur, they often either require costly re-optimization or depend on simplified heuristics that may fail to capture complex interactions between resources and tasks.

While extensive research has advanced resource-constrained project scheduling through various methodologies, such as utilizing genetic programming to discover priority rules for standard task graphs [5], optimizing multi-skill resource constraints with flexible profiles [6], and incorporating spatial transfer times [7], a significant gap remains in the existing literature. Specifically, no prior work evolves per-task, state-dependent allocation rules attached directly to a project task graph for heterogeneous resources that possess both distinct skill sets and specific geographic locations.

Genetic Programming (GP) offers a promising alternative. Instead of computing a full schedule in advance, GP can evolve resource allocation policies, that is, interpretable rules that describe, at each decision point, how to assign available resources at the decision time to current task(s). Such an approach supports dynamic adaptability, generalization for different scenarios, and human-readable decision logic.

In this study, we address the following research questions.

RQ1: Can GP be formulated to support resource allocation for individual tasks in a construction project represented by a task graph?

RQ2: How do the evolved rules compare with a random-allocation baseline on cost and time for a residential project?

RQ3: To what extent do the results obtained across repeated simulation runs indicate the potential benefits and limitations of applying GP to construction resource allocation?

These questions guide the formulation and implementation of the proposed approach and the evaluation of its performance within the considered simulation scenario.

The main contribution of this paper is the development and simulation-based evaluation of a GP-based framework for task-specific resource allocation in construction projects. The framework represents project activities and their precedence relationships using a task graph and applies GP to evolve a resource-selection policy. An important feature of the proposed approach is its consideration of heterogeneous resources, including resources with different skill sets, resources capable of performing multiple types of tasks, and variations in task execution time depending on the resource assigned. This enables resource allocation decisions to account for differences in resource capabilities and productivity rather than treating all resources as interchangeable. The framework is evaluated using a residential-building scenario with 24 tasks and five resources, with results from 30 GP runs compared against 30 runs of a random-allocation baseline. The analysis considers both project cost and completion time, thereby examining the trade-off between these objectives.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature on construction project scheduling, resource allocation, and evolutionary computation, with particular attention to GP and related optimization approaches. Section 3 presents the problem formulation, including the project task graph, resource representation, and objective function. Section 4 describes the proposed GP-based framework and its implementation, including the representation of candidate solutions and the evolutionary operators. Section 5 presents the experimental setup and case study, followed by the simulation results and comparison with the random-allocation baseline. Section 6 discusses the findings, their implications, and the limitations of the study. Finally, Section 7 summarizes the main conclusions and outlines directions for future research.

2. Literature Review

This section reviews three relevant research areas: resource allocation and scheduling in construction problems, evolutionary computation approaches to scheduling and optimization in a general context, and the use of GP for rule discovery and decision-making in dynamic environments.

Efficient resource allocation and scheduling are important elements of construction management. Construction projects typically require the coordination of the assignment of multiple resource types, individual workers, specialized teams, and equipment, to tasks with different skill requirements, durations, and spatial constraints. The complexity of the problem has led to the development of a range of scheduling methodologies. The Critical Path Method (CPM) and the Program Evaluation and Review Technique (PERT) are the main approaches in construction scheduling. CPM identifies the sequence of activities that determines the minimum project duration, assuming deterministic activity times. PERT extends it by adding probabilistic activity durations, enabling planners to account for some uncertainty. Integrating both methods provides a framework that improves schedule robustness and supports more informed decision-making in dynamic environments [8]. However, it is based on the assumption of unlimited resource availability and therefore cannot resolve conflicts arising from limited workers or equipment. As construction projects grow in scale and complexity, these limitations become increasingly restrictive [9]. Another approach used is the Resource-Constrained Project Scheduling Problem (RCPSP), which explicitly uses resource limits in scheduling decisions. Its objective is to minimize project makespan and satisfy precedence and resource constraints. Exact optimization methods, such as mixed-integer programming and branch-and-bound, can solve small instances but scale poorly due to the NP-hard nature of the problem [9,10]. Priority-rule heuristics (e.g., shortest processing time, minimum slack, earliest due date) remain popular due to their computational efficiency. Yet their performance is highly problem-dependent, and no single rule consistently outperforms others across project types [11]. Industry practice has produced domain-specific heuristics such as crew balancing, equipment leveling, and location-based scheduling, which embed practical knowledge of construction workflows. These methods improve applicability but often rely on static assumptions that make it hard to adapt them to changing site conditions [12]. Commercial project management software, such as Microsoft Project, Primavera P6, and Asta Powerproject, implements several resource leveling and RCPSP heuristics. Comparative studies show that while all three tools produce feasible schedules, Asta Powerproject provides the most consistent overall performance, whereas Primavera offers the fastest leveling capabilities [13].

A recurring limitation of these methods is their lack of adaptability. Construction sites are very dynamic; there are delays that can propagate, equipment breakdowns, and resource availability changes. Therefore, static schedules must be frequently recomputed, and simple dispatching rules are often unable to capture complex interactions between tasks and resources.

Evolutionary computation (EC) has been widely applied to scheduling problems due to its flexibility and ability to explore large, nonlinear search spaces. Genetic algorithms (GAs) are the most widely used EC techniques in scheduling. In job shop and flow shop scheduling, chromosomes typically encode task sequences, machine assignments, or priority lists. GAs have demonstrated strong performance in navigating highly constrained search spaces and generating near-optimal schedules for problems that are computationally intractable for exact methods [14,15]. Many real scheduling problems involve many objectives, such as minimizing makespan, reducing cost, and improving resource utilization. Multi-objective evolutionary algorithms (MOEAs), especially NSGA II and SPEA2, have been widely adopted to address these trade-offs. NSGA II's fast nondominated sorting and elitism

enable efficient exploration of Pareto optimal schedules, while SPEA2's strength-based fitness assignment improves convergence and diversity [16,17]. Another approach that was used was based on combining EC with techniques such as tabu search, simulated annealing, or constraint-based reasoning. In scheduling, they have shown superior performance compared to pure evolutionary approaches, particularly in job shop and project scheduling problems [18,19].

In construction management, EC methods have been applied to a range of resource-related problems, such as crew scheduling and allocation, optimizing labor distribution across tasks with multiple skill requirements [20], equipment allocation and fleet management, including crane placement, earthmoving operations, and equipment routing [21], and resource leveling, reducing fluctuations in resource usage to improve cost efficiency and reduce idle time [22]. These applications show that EC methods usually outperform classical heuristics and often achieve near-optimal solutions for static or semi-static scheduling problems.

GP differs from other evolutionary methods by evolving programs or decision rules rather than fixed solutions [23]. This makes GP suitable for dynamic environments where decisions must be made repeatedly based on the current system state. GP has been successfully applied to different problems, including dispatching rule evolution in manufacturing systems [24], resource allocation in logistics, such as vehicle routing [25], adaptive control for multi-agent coordination, where GP evolves policies that respond to changing conditions [26] or different hyper-heuristic frameworks, where GP generates heuristics that generalize across problem instances [27–29].

A key advantage of GP is its interpretability, which means that evolved rules can often be expressed in a human-readable way, enabling domain experts to understand the resulting policies. This is especially important in construction, where transparency and explainability are essential for adoption by practitioners, and are often required by various certification and compliance rules and regulatory boards. Despite these strengths, the research on the application of GP to construction resource allocation is limited. Existing studies primarily focus on static scheduling or narrow subproblems such as equipment routing. Few works explore GP as a means of evolving dynamic, state-dependent allocation policies that operate over a project's task graph [30–32] and adapt to disturbances during execution.

The reviewed literature indicates that while construction scheduling has heavily relied on static heuristics and one-shot optimization models, evolutionary approaches for truly online, dynamic allocation remain scarce. This limitation is highlighted in recent systematic literature review, which concludes that current scheduling paradigms under uncertainty and resource flexibility critically lack adaptive, real-time policy generation mechanisms [33]. Although GP and hyper-heuristics have been widely explored for discovering dispatching rules in manufacturing and general scheduling [24,27,28], with recent advances applying GP hyper-heuristics to dynamic resource-constrained multi-project scheduling, a distinct methodological boundary distinguishes those methods from this work [29]. Specifically, traditional hyper-heuristics and dispatching approaches evolve a single, global priority rule that is applied uniformly across all decision points in the system. In contrast, the approach used here assigns a distinct allocation rule directly to each task node within the project task graph; while this significantly enlarges the search space, it also permits task-specific logic and decentralized adaptation. Furthermore, while data structures like the extended task graph have been conceptualized for general optimization and automotive production processes [30,31], their utilization as a substrate for evolving fine-grained, localized resource-allocation policies in construction, specifically accounting for heterogeneous resources with distinct skills, locations, and resource-specific cost-time dynamics, remains entirely unaddressed.

This paper addresses this gap by proposing a GP-based framework that evolves resource-allocation rules attached to the nodes of a task graph, enabling adaptive decision-making during the whole project execution. The approach combines the interpretability of rule-based systems with the flexibility of evolutionary search, offering a novel contribution to both construction management and evolutionary computation.

3. Problem Definition

Construction projects consist of interdependent tasks that must be executed by different resources, both specialized and universal, within time, spatial, and operational constraints. The goal of dynamic resource allocation is to assign available resources to possible tasks during project execution in a way that minimizes project duration, reduces idle time, and/or maintains efficient use of workers and equipment. This section formalizes the project structure, resource characteristics, decision process, and optimization objectives addressed in this work.

Each construction project is modeled as a directed acyclic graph (DAG) as given by Equation (1).

$$G = (V, E), \tag{1}$$

where $V = \{1, 2, ..., n\}$ is the finite, non-empty set of nodes representing tasks and $E \subseteq V \times V$ is a set of directed edges representing precedence constraints. An edge (i, j) indicates that task j cannot begin until task i is completed.

Each task $i \in V$ has several attributes: duration d_i, required skills or equipment S_i, location L_i, priority or criticality p_i, safety or spatial constraints (e.g., tasks that cannot be executed simultaneously). The DAG structure ensures that all feasible schedules respect the logical and technological order of construction activities.

The resource set is defined by Equation (2)

$$R = R_u \cup R_t, \tag{2}$$

where R_u denotes universal resources, R_t denotes specialized teams. Each resource $r \in R$ has several parameters such as skill set K_r, productivity or speed factor α_r, use cost c_r, mobility or relocation time between locations, availability schedule, including downtime or maintenance. A resource may be assigned to at most one task at a time, and tasks in general may require one or multiple resources depending on their type, although in this paper we assume each task is carried out by a single resource.

In this research, it is assumed that resource allocation is performed at discrete decision points during project execution. At each decision point t, the system observes the current project state consisting of completed tasks, possible tasks whose predecessors are finished, availability and location of each resource, ongoing tasks and their remaining durations, and external conditions such as delays or equipment breakdowns.

The allocation problem consists of selecting, for each eligible task i, a resource r such that:

1. $S_i \subseteq K_r$ (the resource has the required skills),
2. no safety or spatial constraints are violated,
3. the resulting schedule remains feasible with respect to precedence and resource availability.

A complete allocation for the project is obtained by repeatedly applying the allocation policy until all tasks are completed.

The goal is to evolve allocation strategies that optimize one or more performance criteria. The objectives that can be considered are makespan (Equation (3)), use cost (Equation (4)), resource idle time, and robustness.

Makespan (Project Duration).

$$\text{f1} = \min C_{\max}, \tag{3}$$

where $C_{\max}$ is the completion time of the final task.

Use Cost.

$$\text{f2} = \min \sum_{r \in R} c_r \cdot T_r, \tag{4}$$

where T_r is the total active time of resource r.

Resource Idle Time.

Idle time accumulates when resources are available but not assigned to tasks. Minimizing idle time improves utilization and reduces final cost.

Robustness.

Evolved policies should maintain good performance under random delays, equipment breakdowns, and fluctuating resource availability. Robustness would be evaluated by simulating multiple perturbed scenarios. However, robustness has not been evaluated in current experiments.

These objectives may be combined into a weighted fitness function or treated as separate objectives in a multi-objective evolutionary framework. In the example scenario in this paper, several of the metrics are combined for the fitness function.

4. Genetic Programming Framework

This section describes the proposed GP framework for evolving dynamic resource-allocation strategies for construction projects. The approach represents each construction project as a task graph and associates each task with a rule that selects an appropriate resource based on the current project state as well as the resource and task attributes. The resulting allocation policy is evaluated through simulation, and evolutionary operators are applied to improve performance over subsequent generations.

4.1. Representation of Individuals

Each GP individual encodes a complete resource-allocation strategy for the project. The project is represented by the task graph as defined in Equation (1), and each task i is associated with a selection rule RS_i. The selection rule RS_i is evaluated whenever task i is eligible for execution, producing an actual resource selection. This representation ensures that each individual defines a full allocation policy applicable throughout project execution.

4.2. Fitness Evaluation

The fitness of an individual is computed by simulating project execution using the allocation policy encoded by its GP individual. The simulation proceeds in topological order of the task graph, taking into account precedence constraints and resource availability. For each eligible task, the corresponding rule determines the resource assignment.

Once all tasks are scheduled, performance metrics are extracted. The metrics include project makespan $C_{\max}$, total resource idle time, total operational cost, and number of constraint violations (penalized heavily). They are then combined in a weighted fitness function(F), given by Equation (5), to evaluate the quality of an individual, where weights w are set experimentally; for this experiment, they were set, respectively, to 0.6, 0.1, 0.2, and 0.1.

$$F = w_1 C_{\max} + w_2 \cdot \text{IdleTime} + w_3 \cdot \text{Cost} + w_4 \cdot \text{Penalties}, \tag{5}$$

The cost of the system is calculated as the sum of the costs of all the resources used and the sum of the costs of task execution. It is defined by Equation (6) below.

$$Cost = \sum_{i=1}^{m} c_i + \sum_{j=1}^{n} c_j \quad (6)$$

where m is the number of resources and n is the number of tasks in the graph.

4.3. Population Initialization

At the beginning, the initial population is created. The population consists of randomly generated individuals in the form of trees. The root of a single tree is an embryo and is assigned a randomly selected feasible resource to be executed for the first task. The rest of the nodes in the tree represent system construction functions, to which selection rules, listed in Table 1, are assigned. Each selection rule has a probability of being used.

Table 1. Selection rules used and their probabilities.

Selection Rule	Description	Probability
CHP	The cheapest	0.1
FST	The fastest	0.1
MTC	Min (time * cost)	0.3
IDL	Idle for the longest time	0.15
SME	The same as predecessor	0.2
RAR	The rarest used	0.15

Greedy options (the fastest and the cheapest) are assigned the lowest selection probability of 0.1. On the other hand, the multi-criteria option, Min (time * cost), has the highest probability at 0.3. Retaining the predecessor option carries a probability of 0.2, while all remaining options share a probability of 0.15. Finally, the phenotype is constructed by executing functions task by task according to the genotype. After assigning an actual resource to each task, a phenotype is obtained.

4.4. Genetic Operators

Standard GP operators are used. Since they operate on a tree structure rather than a graph, the spanning tree *ST* for a task graph rooted in T_0 is obtained at the beginning of the evolutionary process. Because the proposed methodology belongs to the constructive group of algorithms, the structure of the spanning tree is constant for every individual. The structure is not modified after using genetic operators. It is necessary to ensure that all of the tasks in the task graph will be executed.

Crossover. Two individuals, represented by ST_1 and ST_2, exchange rules associated with a randomly selected subset of tasks, represented by subtrees of *STs*. It is done by selecting a random task T in the ST and then exchanging the subtrees of T between ST_1 and ST_2. As the subtrees differ only in rules assigned to particular tasks, the structure of the resulting offspring remains unchanged.

Mutation. Mutation is performed by replacing a rule in T_i with a randomly selected rule, perturbing numeric constants, or reinitializing the rule for a randomly selected task.

Cloning. The individual is copied to the next population without changes.

The spanning tree (ST) is constructed from the underlying task graph (DAG) by ensuring that for each node with multiple predecessors, only a single incoming edge is retained; in our study, the first one, starting from a designated root node (T_0). During evolutionary optimization, crossover operations swap the specific rules attached to the tasks rather than altering the raw task order; because precedence constraints are dynamically

enforced by the simulator during schedule generation, every offspring is guaranteed to be precedence-feasible by construction. Furthermore, reconvergent paths, where multiple dependency routes diverge and later merge at a shared descendant, cause no structural problems because the tree-building process resolves any ambiguity by keeping only a single parent connection per node, effectively eliminating redundant conflicts while preserving the integrity of the task dependencies.

4.5. Algorithm Overview

The overall evolutionary process described above in Sections 4.1–4.4 can be summarized as follows:

1. The population is initialized with individuals, each containing a selection rule for every task.
2. Project execution is simulated, and fitness is computed for each individual.
3. Individuals are selected for reproduction using tournament selection.
4. Crossover and mutation are applied to generate offspring.
5. Offspring are inserted into the population.
6. This process is repeated until a termination criterion is met.

The final output of the evolutionary process is an evolved resource-allocation policy that can be applied dynamically during project execution. The schematic diagram of the GP Framework is depicted in Figure 1.

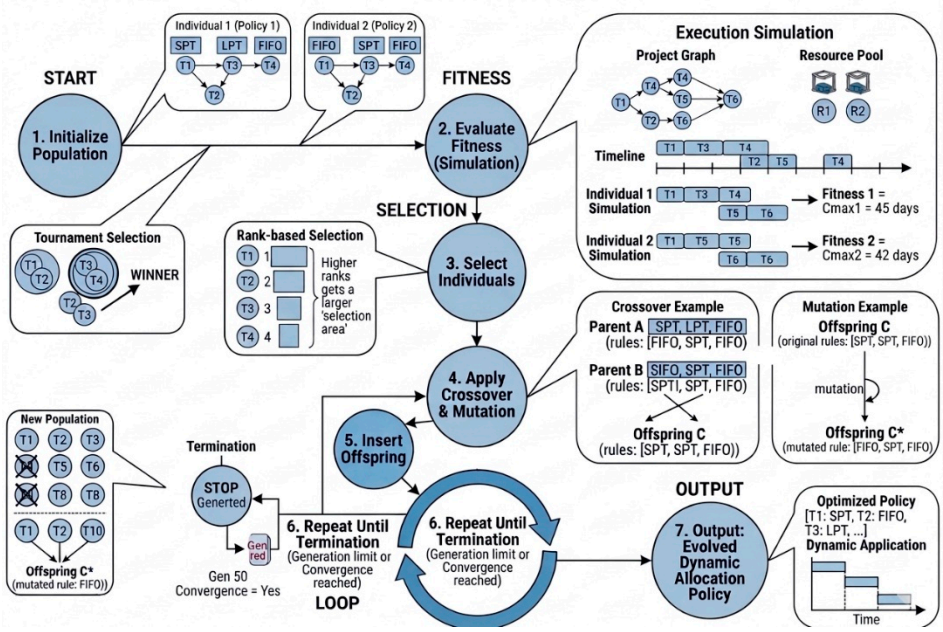


Figure 1. A schematic depiction of the GP Framework for policy generation.

5. Experimental Setup and Results

This section describes the experimental environment used to evaluate the proposed GP framework for dynamic resource allocation in construction projects. The experiments have been conducted using a discrete-event simulation of project execution, a real-world scenario for residential building, and a baseline allocation strategy for comparison. Performance is assessed using standard scheduling metrics.

The experiments were performed using a computer with Windows 11 operating system, 12th gen Intel core 1255U (1.7 GHz) processor and 32 GB RAM memory.

5.1. Simulation Environment

A custom discrete-event simulator is used to model the execution of construction projects. The simulator processes events such as task start, task completion, and resource release. At each decision point, the GP-derived allocation policy is used to assign available resources to eligible tasks. The simulator enforces precedence constraints defined by the task graph, resource availability, and exclusivity. Time advances to the next event, and the process continues until all tasks are completed. A scenario used for experiments is represented as a task graph with task attributes and resource requirements derived from typical construction workflows.

At the core of the event model, the simulation tracks precise timestamps for task starts, task completions, and resource releases, updating the global system state dynamically as time progresses. At each decision point, the state representation encapsulates the current execution status of the project graph (including completed, active, and eligible tasks), remaining resource capacities, and active allocation constraints. The GP-derived allocation rule is evaluated event-drivenly, that is, specifically, whenever a resource becomes free and one or more pending tasks become eligible for execution, matching available resources to competing demands in real time. In these reported experiments, task durations are modeled as deterministic values, and no dynamic disturbances or stochastic disruptions (such as resource breakdowns or unexpected delays) were introduced, establishing a controlled baseline environment to isolate and evaluate the pure performance of the allocation policies.

5.2. Genetic Programming Parameters

The GP algorithm is configured using standard parameter settings. In the experiments, the population size of 200 individuals was used, and the number of generations was set to 100. The selection method used was tournament selection (size 30); the crossover probability was set to 0.7, the mutation probability to 0.2, and cloning to 0.1. The size of the population and the number of individuals were set after a preliminary run, while the genetic operators' probabilities are based on typical values used in the literature. The experiment has been repeated over 30 independent runs to account for stochastic variability.

The experimental scenario models a small residential building project. The dataset has been manually constructed by the authors, based on real-world workflows to accurately reflect practical scheduling constraints. It consists of 24 discrete tasks distributed across ten distinct zones or locations (labeled A through J). Task durations range between 1 and 3 time units, reflecting realistic execution windows for localized construction projects. The project integrates a diverse set of specialized and general trades, ranging from site preparation (surveying, excavation, general site work) and structural phases (concrete, carpentry, roofing) to systems installation (plumbing, electrical), interior finishing (flooring, tiling, painting), and final site work (landscaping and cleanup). Each task is assigned a specific operational priority level (where 1 denotes high-priority or critical path milestone activities like key inspections and structural foundations, 2 represents standard secondary trades, and 3 covers routine finishing, painting, and cleanup tasks), providing a rich hierarchical structure for evaluating resource allocation policies under spatial and skill constraints. All the tasks with parameters are depicted in Table 2 and the task graph representing this example is shown in Figure 2.

In the simulation model, task duration represents the nominal time required to complete an activity under a baseline resource speed of 1.0. Resource performance is governed by a resource-specific speed parameter rather than a task-specific one, where a speed of 1.0 denotes a standard work pace, values greater than 1.0 represent faster execution, and values lower than 1.0 indicate slower performance. Consequently, the actual execution time of a task is determined by scaling its nominal duration relative to the assigned resource's

operational speed multiplier. In general, universal resources are capable of performing a given task at its nominal time or shorter, while specialized resources can execute a task faster. For multi-skilled resources, the speed factor usually is different for different skills.

Table 2. Task parameters for the small residential building project benchmark scenario (compact layout).

i	d_i	S_i	L_i	P_i	i	d_i	S_i	L_i	P_i
0	2	general	A	1	12	2	general	F	2
1	3	excavation	A	1	13	3	carpentry	F	2
2	1	survey	A	2	14	1	inspection	F	1
3	2	concrete	B	1	15	2	flooring	G	3
4	2	carpentry	B	2	16	2	tiling	G	3
5	3	plumbing	C	2	17	1	general	G	3
6	2	electrical	C	2	18	2	landscaping	H	3
7	1	general	C	3	19	1	inspection	H	1
8	2	roofing	D	3	20	2	general	I	2
9	1	inspection	D	1	21	2	painting	I	3
10	2	painting	E	3	22	1	finishing	I	3
11	1	finishing	E	3	23	1	cleanup	J	3

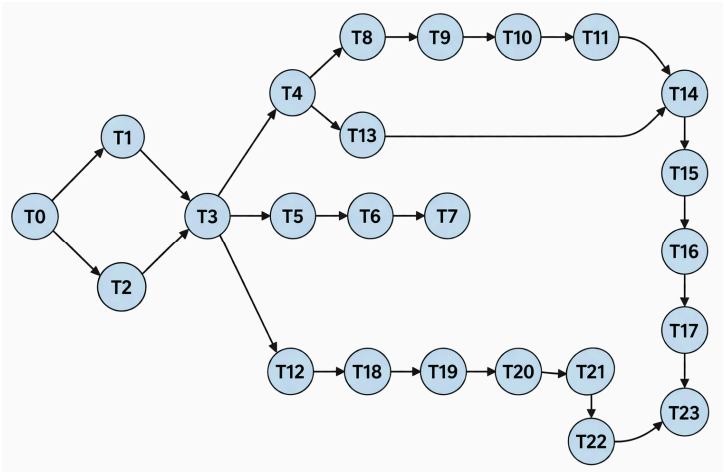


Figure 2. A DAG representing a sequence of tasks with precedence (for a construction scenario for residential building).

Each individual is represented as a spanning tree with selection rules assigned to nodes. An example of such a tree for the experimental scenario is depicted in Figure 3.

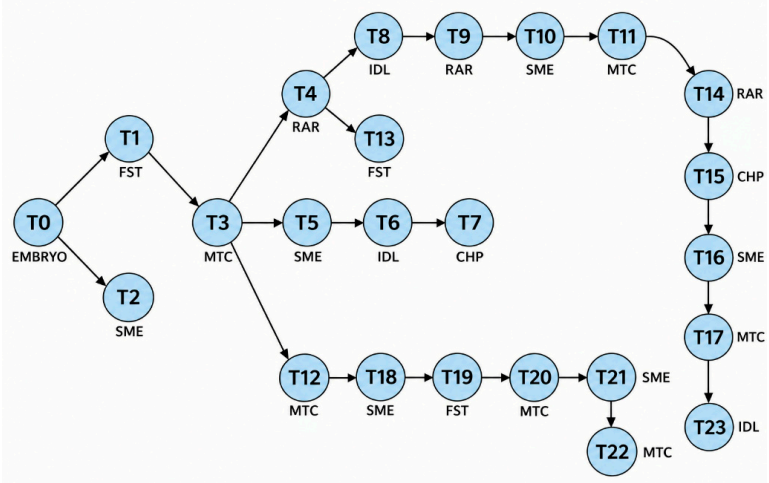


Figure 3. A ST representing an individual (genotype).

This individual represents a resource allocation policy for the 24-task problem. During the simulation, it has been converted to a phenotype by assigning actual resources to each task. The assignment is represented by the following vector: [1, 4, 1, 4, 2, 4, 1, 2, 0, 0, 0, 0, 4, 3, 3, 0, 0, 0, 4, 3, 2, 2, 4, 1]. For this phenotype, the final cost of the project was equal to 3459 simulator cost units. All the tasks finished their execution before 38,358 simulator-unit time.

5.3. Results

To evaluate the efficiency of the proposed solution, we conducted a series of experiments using a residential building project. The performance metrics are summarized in Table 3, which reports the best, worst, and average results obtained across 30 independent runs, in terms of cost and time, respectively. The Generation column indicates the generation in which the corresponding result was found.

Table 3. Comparison of GP and the random baseline over 30 independent runs (cost in simulator units, time in simulator units).

	GP			Random	
	Cost (In Simulator Cost Units)	**Time (In Simulator Time Units)**	**Generation**	**Cost (In Simulator Cost Units)**	**Time (In Simulator Time Units)**
best	813	5950	42	2025	4204
worst	2565	5283	40	3884	4896
average	1157.57	5678.17	n/a	3085	4508.6

The best recorded cost was 813, while the worst-case result reached 2565, indicating a significant spread in performance. However, the mean result (1157.57) suggests that these extremes are likely incidental outliers rather than representative outcomes. Such variance is common in evolutionary algorithms due to their stochastic nature. Consequently, Table 4 provides a comprehensive statistical evaluation. The analysis utilizes the median as a measure of central tendency, alongside several measures of dispersion, including the interquartile range (IQR), sample and population variance, standard deviation, and median absolute deviation (MAD).

Table 4. Statistical measures.

Measure	**Value**
sample variance	167,371.36
population variance	161,792.31
standard deviation	409.11
IQR	204.75
median	1049
median absolute deviation	93.5
Mann-Whitney U-test (U-value)	22
Mann-Whitney U-test (p-value)	2.60×10^{-10}

As expected, the sample variance (167,371.36), population variance (161,792.31), standard deviation (409.11), and interquartile range (204.75) all indicated a substantial spread in the results. To determine if this dispersion was caused by occasional outliers, the MAD was calculated. Because the MAD is robust against extreme values, it serves to clarify whether outliers were incidental or symptomatic of the broader distribution. The resulting MAD of 93.5, relative to the median (1049), suggests that extreme outcomes were indeed incidental rather than frequent, but it needs further experiments to fully confirm. The Mann-Whitney U test revealed a significant difference between the algorithm presented in

this paper and the random solution. This was confirmed by a low U-value (equal to 22) and p-value (2.60×10^{-10}).

The best cost produced by the random solution was equal to 2025. However, the average result from 30 tries was equal to 3085. The worst result obtained using the random algorithm has a cost equal to 3884. The graphical comparison of the results is presented in Figure 4. It depicts the values of cost in simulator cost units.

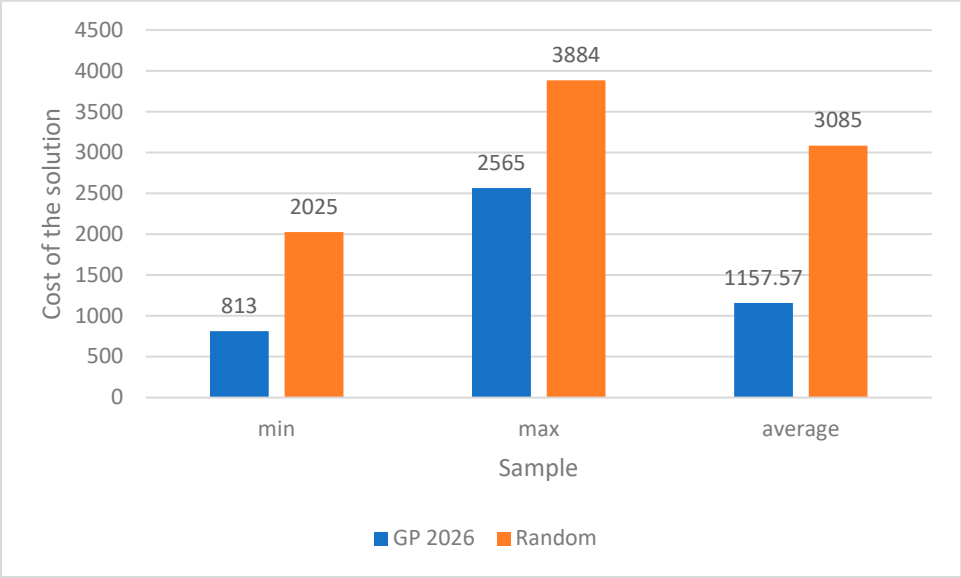


Figure 4. A comparison of performance between the random baseline and GP Framework approach.

As observed, our proposed algorithm significantly outperforms the random-allocation baseline for this specific project scenario. Because the baseline serves as a lower reference point rather than a competitive heuristic, both its best and average performance metrics were several times worse than those obtained using our approach.

The results indicate that GP can discover allocation strategies that balance resource utilization, task prioritization, and spatial considerations more effectively than static heuristics.

An important advantage of GP is the interpretability of the evolved allocation rules. Several high-performing individuals contained concise expressions that captured intuitive domain knowledge, such as prioritizing tasks on the critical path, assigning resources with matching skills and minimal relocation distance, or delaying low-priority tasks when high-cost machinery is available. These interpretable rules may support adoption in real-world construction management settings, where transparency and explainability are essential. These findings highlight the potential of GP as a powerful tool for dynamic resource allocation in construction projects.

6. Discussion

The results provide an initial assessment of GP as an approach to resource allocation in construction projects. The proposed framework represents a construction project as a DAG and associates tasks with resource-selection rules that consider task and resource characteristics. This representation is particularly relevant to construction environments, where activities have precedence constraints and resources differ in their capabilities, skill sets, operating costs, and task execution times. Unlike allocation strategies that treat resources as interchangeable, the proposed framework can account for these differences when selecting a resource for an eligible task. This differs from GP hyper-heuristic studies on the resource-constrained project scheduling problem, which evolve priority rules intended

to generalize across project instances [5,29,34]. In the present framework, the rules are attached to the individual tasks within a specific project, which can capture task-specific allocation logic but may enlarge the search space. General overviews of the automated design of scheduling heuristics with GP are given in [35,36]. The results suggest that evolving task-specific allocation rules is a promising direction for exploring resource assignments in such environments. However, the findings can only be interpreted in the context of the experimental design and the single residential-building case study considered in this work.

The comparison between GP and the random-allocation baseline shows a difference in the cost and time outcomes. Across 30 independent runs, GP achieved a mean cost of 1157.57 simulator cost units, compared with 3085 units for the random baseline. This corresponds to a reduction of approximately 62.5% in the mean simulated cost. The best GP result was 813 units, whereas the best random-allocation result was 2025 units. These results indicate that the GP search can identify allocation strategies with substantially lower simulated costs than those produced by random allocation in the considered scenario. Nevertheless, this cost reduction does not translate into an improvement in every performance measure. The mean completion time for GP was 5678.17 simulator time units, compared with 4508.6 units for the baseline. Thus, GP achieved lower mean cost at the expense of a longer mean completion time. This trade-off is important for construction management because the preferred allocation strategy depends on project priorities: minimizing resource expenditure may be more important in some situations, while meeting a strict completion deadline may take precedence in others. Consequently, the results do not demonstrate that GP is universally superior to alternative scheduling approaches; rather, they illustrate its potential to produce different cost–duration trade-offs. Direct numerical comparison with published results is not possible. GP-based studies on the resource-constrained project scheduling problem typically benchmark evolved rules against established priority rules on standard instance sets, with project duration as the main performance measure [5,34,37]. The present study instead considers a realistic project with skill requirements, locations, and resource-specific costs and durations, which are outside the basic resource-constrained project instance sets [38], and it reports cost and time in simulator units. The approximately 62.5% difference in mean cost is therefore not comparable to the improvements reported in those studies. In the construction domain, reviews classify the available scheduling optimization methods as mathematical, heuristic, and metaheuristic [39] and report genetic algorithms as the most frequently applied technique [40]. Earlier GA-based studies addressed resource allocation, resource leveling, and equipment selection [20–22]. Evolving allocation rules rather than a fixed schedule is the distinguishing feature of the present approach, but a full comparison with such methods remains to be carried out.

The observed variation between GP runs also highlights the stochastic nature of evolutionary search. The reported mean GP cost is higher than the median of 1049 units, and the cost distribution shows dispersion, as indicated by the standard deviation of 409.11 and the interquartile range of 204.75. The median absolute deviation is 93.5 units. These descriptive statistics provide complementary information about the distribution of outcomes, but they are not sufficient to establish that the most extreme observations are incidental outliers. The repeated runs provide a useful initial view of variability under the chosen experimental configuration; however, stronger comparative conclusions would require analysis of the complete run-level data, confidence intervals, and appropriate statistical tests. Nonparametric tests, which do not assume normally distributed results, are a standard choice for comparing stochastic search algorithms over independent runs [41]. The results should therefore be understood as descriptive evidence of the method's behavior in the studied instance.

The findings also have implications for the design of resource-allocation methods. GP offers a flexible way to search for task-specific rules without requiring the allocation policy to be defined entirely in advance. This flexibility is potentially valuable when tasks differ in skill requirements, resources have multiple capabilities, and assigning different resources to the same task can result in different execution times and costs. At the same time, the quality of the resulting policy depends on the representation of candidate rules, the fitness function, the evolutionary operators, and the information available to the decision process. The weighting of cost, completion time, and other criteria can influence which allocation strategies are favored during evolution. A policy optimized under one weighting scheme may not be appropriate when project priorities change. Further investigation should therefore examine how the fitness-function weights affect the resulting schedules and whether a multi-objective formulation could provide a clearer representation of the trade-offs between cost, duration, and resource utilization. Multi-objective evolutionary algorithms, such as NSGA-II [16] and SPEA2 [17], provide established means of obtaining sets of non-dominated solutions instead of a single weighted-sum solution, and multi-objective GP has been used to evolve project scheduling heuristics while controlling their complexity [34].

Although the framework is motivated by dynamic resource allocation, an important distinction must be made between the design of a policy intended for repeated decisions during execution and demonstrated adaptation to unforeseen disturbances. The current case study does not provide a systematic evaluation under simulated delays, equipment failures, or changes in resource availability. Accordingly, robustness to such disturbances cannot be concluded from the reported results. Dynamic project scheduling studies consider such disruptions explicitly: GP-evolved priority heuristics have been tested on instances with changing resource availability [34], dual-tree GP has been used to evolve separate rules for normal and disrupted states [42], and scheduling under uncertainty has been reviewed in [33,43,44]. Evaluation against such disruptions is the main element missing from the present study. Similarly, although GP can produce rules that may be inspected as expressions, interpretability and practical usability have not yet been established through a systematic analysis of evolved rules or evaluation by construction practitioners. Rule complexity has been treated as an explicit concern in GP-based scheduling, either as a second objective [34] or through analysis of the size of the evolved rules [45]. These properties should be treated as potential advantages of the approach that require further validation.

Limitations

Several limitations must be considered when interpreting the results. First, the experimental evaluation uses a single residential-building scenario consisting of 24 tasks and five resources. Although this case represents a realistic setting for evaluating the proposed framework, it does not represent the diversity of construction projects, which can differ considerably in scale. The results therefore cannot be generalized to other projects without further evaluation. In addition, the task graph and resource attributes determine which allocation decisions are feasible and how their costs and durations are assessed. The completeness and representativeness of these inputs are important to the validity of the conclusions.

Second, the baseline comparison is limited to random allocation. Although this baseline provides a reference for assessing whether the GP search can find better solutions than unstructured allocation, it is not sufficient to establish an advantage over established scheduling and optimization techniques. Priority-rule heuristics, such as shortest processing time, minimum slack, and critical-path-based rules, would provide more informative comparisons. Such comparisons would help distinguish improvements arising from evolu-

tionary search from those attributable to the specific representation and fitness function used in this study.

Third, the experimental evaluation is based on a weighted fitness function. The selected coefficients influence the relative importance assigned to the performance criteria and may affect which allocation strategies are favored. Without a systematic sensitivity analysis, it is not possible to determine how stable the results are under alternative weight configurations. As future work, we plan to evaluate alternative weight settings.

Fourth, the number of independent runs and the descriptive statistics provide an initial characterization of variability, but the analysis does not establish statistical significance or quantify uncertainty in the differences between methods.

Fifth, the reported cost and time values are expressed in simulator units and have not been calibrated against actual project expenditure or calendar time. The approximately 62.5% difference in mean cost therefore represents a relative improvement within the simulation model, not a reduction in monetary expenditure on the construction project.

Finally, the framework's intended applicability to dynamic construction environments must be tested more comprehensively. The current experiments do not evaluate policy behavior under disturbances, and the practical interpretability of evolved rules has not been established through expert assessment. Future studies should evaluate multiple project instances, introduce controlled disturbances, assess the stability of allocation decisions, inspect the complexity of evolved rules, and involve construction professionals in judging their validity and usefulness.

Overall, this study should be viewed as an initial simulation-based investigation of GP for resource allocation in a construction project. The results show promising cost outcomes relative to a random baseline, while also revealing a trade-off with completion time. Further validation against established scheduling methods, across additional project instances, and under controlled dynamic conditions is required before drawing broader conclusions about practical effectiveness.

7. Conclusions

This paper introduced a GP framework for dynamic resource allocation in construction projects. Unlike traditional scheduling and heuristic-based approaches, the proposed method evolves allocation policies that operate directly on the structure of a project's task graph rather than static allocation. Each GP individual encodes a set of decision expressions associated with tasks, enabling adaptive and context-aware resource assignment throughout project execution. The experimental results on a real-world residential building construction project demonstrate that the GP-evolved policies outperform a random allocation heuristic. Improvements were observed in project makespan, resource idle time, and operational cost.

An additional benefit of the proposed approach is the interpretability of the evolved allocation rules. Many high-performing individuals contained concise, human-readable expressions that align with intuitive construction management principles. This interpretability supports practical adoption, as decision-makers can inspect, validate, and refine the evolved policies.

Future work will explore several extensions of the proposed framework. In this paper, a small-sized scenario has been used, but to evaluate the scalability of the proposed framework, the approach will be evaluated for larger construction projects with more complex task graphs derived from real-world cases. Moreover, to avoid the need for spanning tree generation, we plan to adapt genetic operators to the graph structure of the problem. In addition, as the selection rules associated with the tasks can be written in the form of mathematical expressions, we plan to experiment with representing them by expression trees and evolving by applying GP operators, thus moving to a form of hierarchical genetic

programming framework. Future work will also include integrating multi-objective evolutionary algorithms to explicitly balance competing performance criteria, incorporating richer spatial and safety constraints, and coupling the GP-based policies with Building Information Modeling (BIM) systems for real-time deployment.

Overall, the results indicate that GP offers a powerful and flexible approach for dynamic resource allocation in construction projects, providing both performance improvements and interpretable decision logic suitable for real-world applications.

Author Contributions: Conceptualization, B.S., J.S. and A.G.; Methodology, B.S. and J.S.; Software, B.S. and A.G.; Validation, J.S., B.S. and A.G.; Formal Analysis, B.S.; Investigation, B.S., J.S. and A.G.; Resources, B.S.; Data Curation, B.S. and A.G.; Writing—Original Draft Preparation, B.S., J.S. and A.G.; Writing—Review and Editing, B.S., J.S. and A.G.; Visualization, B.S., J.S. and A.G.; Supervision, B.S. All authors have read and agreed to the published version of the manuscript.

Funding: The authors have not received any external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available from the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this work, the authors used free versions of ChatGPT and Grammarly in order to improve the readability and language of the manuscript. Moreover, the authors have used Copilot to improve the quality of the workflow diagram in Figure 2. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BIM	Building Information Modeling
CPM	Critical Path Method
DAG	Directed Acyclic Graph
EC	Evolutionary Computation
GA	Genetic Algorithm
GDP	Gross Domestic Product
GP	Genetic Programming
IQR	Interquartile range
MAD	Median absolute deviation
MOEA	Multi-Objective Evolutionary Algorithm
PERT	Program Evaluation and Review Technique
RCPSP	Resource-Constrained Project Scheduling Problem
ST	Spanning tree

References

1. OECD. *Due Diligence for Responsible Sand and Silicate Supply Chains*; OECD Publishing: Paris, France, 2026. [CrossRef]
2. McKinsey Global Institute. *Reinventing Construction: A Route to Higher Productivity*; McKinsey & Company: New York, NY, USA, 2017; Available online: https://www.mckinsey.com (accessed on 25 October 2026).
3. Sambasivan, M.; Soon, Y.W. Causes and effects of delays in Malaysian construction industry. *Int. J. Proj. Manag.* **2007**, *25*, 517–526. [CrossRef]
4. Amoah, K.B.O.; Okere, G.; Deshpande, A. Construction delay analysis: Causes, impacts, and mitigation strategies. *J. Civ. Eng. Res.* **2024**, *14*, 1–9. [CrossRef]
5. Luo, J.; Vanhoucke, M.; Coelho, J.F.d.S.; Guo, W. An efficient genetic programming approach to design priority rules for resource-constrained project scheduling problem. *Expert Syst. Appl.* **2022**, *198*, 116753. [CrossRef]

6. Luo, X.; Guo, S.; Du, B.; Luo, X.; Guo, J. Multi-objective multi-skill resource-constrained project scheduling considering flexible resource profiles. *Appl. Sci.* **2024**, *14*, 1921. [CrossRef]
7. Hosseinian, A.H.; Baradaran, V. A multi-objective multi-agent optimization algorithm for the multi-skill resource-constrained project scheduling problem with transfer times. *RAIRO Oper. Res.* **2021**, *55*, 2093–2128. [CrossRef]
8. Itani, K. Mastering construction schedules: The power of CPM and PERT integration. *Int. J. Res. Appl. Sci. Eng. Technol.* **2023**, *11*. [CrossRef]
9. Chaudhary, A.; Meshram, S. Comparison of various techniques for project scheduling under resource constraints. *Int. J. Eng. Trends Technol.* **2025**, *73*, 57–72. [CrossRef]
10. Brucker, P.; Drexl, A.; Möhring, R.; Neumann, K.; Pesch, E. Resource-constrained project scheduling: Notation, classification, models, and methods. *Eur. J. Oper. Res.* **1999**, *112*, 3–41. [CrossRef]
11. Kolisch, R.; Hartmann, S. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In *Project Scheduling: Recent Models, Algorithms and Applications*; Weglarz, J., Ed.; Springer: Boston, MA, USA, 1999; pp. 147–178. [CrossRef]
12. Isaac-Laughter, O.; Ogunnubi, T.; Orimoloye, M.; Abraham, A.; Ukwunna, C. Project scheduling tools and techniques for effective construction management. *Adv. Multidiscip. Sci. Res. J.* **2019**, *15*, 321–338. [CrossRef]
13. Albayati, N.H.F.; Aminbakhsh, S. Resource allocation capabilities of commercial project management software packages for resource leveling and resource constrained project scheduling problem: A comparative study. *J. Constr. Eng. Manag. Innov.* **2023**, *6*, 104–123. [CrossRef]
14. Hartmann, S. A competitive genetic algorithm for resource-constrained project scheduling. *Nav. Res. Logist.* **1998**, *45*, 733–750. [CrossRef]
15. Gen, M.; Cheng, R. *Genetic Algorithms and Engineering Optimization*; Wiley: New York, NY, USA, 2000. [CrossRef]
16. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
17. Zitzler, E.; Laumanns, M.; Thiele, L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*; Technical Report 103; Computer Engineering and Networks Laboratory (TIK), ETH Zurich: Zurich, Switzerland, 2001. [CrossRef]
18. Ong, Y.S.; Lim, M.H.; Zhu, N.; Wong, K.W. Classification of adaptive memetic algorithms: A comparative study. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2006**, *36*, 141–152. [CrossRef] [PubMed]
19. Ruiz, R.; Stützle, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *Eur. J. Oper. Res.* **2007**, *177*, 2033–2049. [CrossRef]
20. Zhang, H.; Li, H. Simulation-based optimization for dynamic resource allocation in construction. *Autom. Constr.* **2004**, *13*, 409–420. [CrossRef]
21. Marzouk, M.; Moselhi, O. Selecting earthmoving equipment fleets using genetic algorithms. In *Proceedings of the Winter Simulation Conference*; IEEE: Piscataway, NJ, USA, 2003; pp. 1789–1796. [CrossRef]
22. Hegazy, T. Optimization of resource allocation and leveling using genetic algorithms. *J. Constr. Eng. Manag.* **1999**, *125*, 167–175. [CrossRef]
23. Anselmi, F.; Castelli, M.; d'Onofrio, A.; Manzoni, L.; Mariot, L.; Saletta, M. Local search, semantics, and genetic programming: A global analysis. *Soft Comput.* **2026**, *30*, 1541–1559. [CrossRef]
24. Zeiträg, Y.; Figueira, J.R. Automatically evolving preference-based dispatching rules for multi-objective job shop scheduling. *J. Sched.* **2023**, *26*, 289–314. [CrossRef]
25. Gil-Gala, F.J.; Đurasević, M.; Jakobović, D.; Varela, R. Genetic programming with surrogate evaluation for the electric vehicle routing problem. *Swarm Evol. Comput.* **2025**, *96*, 101969. [CrossRef]
26. Naeini, A.T.; Palhang, M. Evolving a multiagent coordination strategy using Genetic Network Programming for pursuit domain. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation, Hong Kong, China, 1–6 June 2008; pp. 3102–3107. [CrossRef]
27. Burke, E.K.; Hyde, M.R.; Kendall, G.; Ochoa, G.; Özcan, E.; Woodward, J.R. Exploring hyper-heuristic methodologies with genetic programming. In *Computational Intelligence: Collaboration, Fusion and Emergence*; Mumford, C.L., Jain, L.C., Eds.; Intelligent Systems Reference Library; Springer: Berlin/Heidelberg, Germany, 2009; Volume 1, pp. 177–201. [CrossRef]
28. Burke, E.K.; Gendreau, M.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; Qu, R. Hyper-heuristics: A survey of the state of the art. *J. Oper. Res. Soc.* **2013**, *64*, 1695–1724. [CrossRef]
29. Chao, Y.; Zhuang, C.; Guo, H.; Liu, J. A genetic programming hyper-heuristic with whale optimization algorithm for the dynamic resource-constrained multi-project scheduling problems. *Expert Syst. Appl.* **2026**, *295*, 128881. [CrossRef]
30. Górski, A.M. Extended task graph for real-life optimization problems. In *Proceedings of the International Conference on the Dynamics of Information Systems*; Springer Nature: Cham, Switzerland, 2025; pp. 74–86.
31. Górski, A.M. Optimization of a car production process specified by an extended task graph. In *Dynamics of Information Systems*; Springer Nature: Cham, Switzerland, 2026; pp. 163–174.

32. Sapiecha, K.; Ciopiński, L.; Deniziak, S. An application of developmental genetic programming for automatic creation of supervisors of multi-task real-time object-oriented systems. In *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS)*; Ganzha, M., Maciaszek, L.A., Paprzycki, M., Eds.; IEEE: Piscataway, NJ, USA, 2014; pp. 501–509. [CrossRef]
33. Aghileh, M.; Tereso, A.; Alvelos, F.; Monteiro Lopes, M.O. Multi-project scheduling under uncertainty and resource flexibility: A systematic literature review. *Prod. Manuf. Res.* **2024**, *12*, 2319574. [CrossRef]
34. Chand, S.; Singh, H.; Ray, T. Evolving heuristics for the resource constrained project scheduling problem with dynamic resource disruptions. *Swarm Evol. Comput.* **2019**, *44*, 897–912. [CrossRef]
35. Branke, J.; Nguyen, S.; Pickardt, C.W.; Zhang, M. Automated design of production scheduling heuristics: A review. *IEEE Trans. Evol. Comput.* **2016**, *20*, 110–124. [CrossRef]
36. Nguyen, S.; Mei, Y.; Zhang, M. Genetic programming for production scheduling: A survey with a unified framework. *Complex Intell. Syst.* **2017**, *3*, 41–66. [CrossRef]
37. Kolisch, R.; Hartmann, S. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *Eur. J. Oper. Res.* **2006**, *174*, 23–37. [CrossRef]
38. Hartmann, S.; Briskorn, D. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2022**, *297*, 1–14. [CrossRef]
39. Zhou, J.; Love, P.E.D.; Wang, X.; Teo, K.L.; Irani, Z. A review of methods and algorithms for optimizing construction scheduling. *J. Oper. Res. Soc.* **2013**, *64*, 1091–1105. [CrossRef]
40. Faghihi, V.; Nejat, A.; Reinschmidt, K.F.; Kang, J.H. Automation in construction scheduling: A review of the literature. *Int. J. Adv. Manuf. Technol.* **2015**, *81*, 1845–1856. [CrossRef]
41. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
42. Chen, H.; Li, X.; Gao, L. A surrogate-assisted dual-tree genetic programming framework for dynamic resource constrained multi-project scheduling problem. *Int. J. Prod. Res.* **2024**, *62*, 5631–5653. [CrossRef]
43. Herroelen, W.; Leus, R. Project scheduling under uncertainty: Survey and research potentials. *Eur. J. Oper. Res.* **2005**, *165*, 289–306. [CrossRef]
44. Demeulemeester, E.; Herroelen, W. Robust project scheduling. *Found. Trends Technol. Inf. Oper. Manag.* **2011**, *3*, 201–376. [CrossRef]
45. Luo, J.; Vanhoucke, M.; Coelho, J. Automated design of priority rules for resource-constrained project scheduling problem using surrogate-assisted genetic programming. *Swarm Evol. Comput.* **2023**, *81*, 101339. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.