buildings

MDPI

Article

A Memetic Algorithm for the Solution of the Resource Leveling Problem

Mehdi Iranagh [1], Rifat Sonmez [2], Tankut Atan [3], Furkan Uysal [4,*] and Önder Halis Bettemir [5]

[1] Transit & Rail Company, Toronto, ON 43964, Canada; mairanagh@gmail.com
[2] Civil Engineering Department, Middle East Technical University, Ankara 06531, Türkiye; rsonmez@metu.edu.tr
[3] Department of Industrial Engineering, Bahçeşehir University, Istanbul 34353, Türkiye; sabritankut.atan@eng.bahcesehir.edu.tr
[4] College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait
[5] Department of Civil Engineering, İnönü University, Malatya 44280, Türkiye; onder.bettemir@inonu.edu.tr
* Correspondence: furkan.uysal@aum.edu.kw

Abstract: In this paper, we present a novel memetic algorithm (MA) for the solution of the resource leveling problem (RLP). The evolutionary framework of the MA is based on integration of a genetic algorithm and simulated annealing methods along with a resource leveling heuristic. The main objective of the proposed algorithm is to integrate complementary strengths of different optimization methods and incorporate the individual learning as a separate process for achieving a successful optimization method for the RLP. The performance of the MA is compared with the state-of-the-art leveling methods. For small instances up to 30 activities, mixed-integer linear models are presented for two leveling metrics to provide a basis for performance evaluation. The computational results indicate that the new integrated framework of the MA outperforms the state-of-the-art leveling heuristics and meta-heuristics and provides a successful method for the RLP. The limitations of popular commercial project management software are also illustrated along with the improvements achieved by the MA to reveal potential contributions of the proposed integrated framework in practice.

Keywords: resource leveling; project scheduling; optimization; genetic algorithms; simulated annealing; memetic algorithms

Citation: Iranagh, M.; Sonmez, R.; Atan, T.; Uysal, F.; Bettemir, Ö.H. A Memetic Algorithm for the Solution of the Resource Leveling Problem. *Buildings* **2023**, *13*, 2738. https://doi.org/10.3390/buildings13112738

Academic Editor: Maziar Yazdani

Received: 9 September 2023
Revised: 20 October 2023
Accepted: 24 October 2023
Published: 30 October 2023

1. Introduction

The critical path method (CPM) is used extensively for the scheduling of construction projects. The CPM performs scheduling by only considering precedence relationships between the activities. However, the CPM does not optimize resources during scheduling; hence, the resource demand diagrams obtained by the CPM usually include undesirable fluctuations. Fluctuations in the manpower and machinery diagrams can be quite expensive because they often cause extra labor or financial expenditure [1–3]. The irregularities in the resource diagram can be leveled by shifting the non-critical activities through their floats.

The resource leveling problem (RLP) aims to determine the optimal schedule so that the fluctuations in the resource diagram are minimized based on a leveling metric. Resource leveling is crucial for optimal planning of construction resources, particularly manpower and machinery, to minimize project costs. Despite the importance of resource leveling in practice, commercial project management software utilizes simple priority-based heuristics and has very limited capabilities for solving the resource leveling problem [4,5]. Hence, development of effective optimization methods for resource leveling has both theoretical and practical relevance.

The methods proposed for the RLP could be categorized into three areas: exact methods, heuristics, and meta-heuristics. The RLP is non-deterministic polynomial-time

hard (NP-hard) in the strong sense [6], and as the problem size grows, the amount of time it takes to solve the problem increases exponentially. Hence, exact methods based on integer-linear programming [2,7–12], dynamic programming [13], branch and bound [14,15], and complete enumeration [16] methods can only solve problems including few activities. In a recent study of scheduling problems subject to general temporal constraints, instances up to 50 activities and five resources were solved to optimality [10].

Numerous heuristic procedures have been proposed in the literature for the RLP. The majority of the heuristic methods used simple shifting heuristics and shifting heuristic priority-rule techniques [17–20]. Case examples up to 12 activities were included to validate the heuristic methods; however, computational experiments were not implemented for performance evaluation in the majority of heuristic studies. Few studies focused on evaluating the capabilities of the project management software for the RLP. Son and Mattila [4] used a problem consisting of 11 activities to reveal the limitations of SureTrak Project Manager version 3.0 and Primavera Project Planner (P3) version 3.0. Iranagh and Sonmez [5] compared the performance of a sole GA with the performance of Microsoft Project 2010 for leveling problems including up to 20 activities with a single resource type.

There has been an increasing interest in the adaptation of meta-heuristics to the RLP in recent years. GAs [5,21–28], neural networks [29,30], particle swarm optimization [31], ant colony optimization [32], artificial bee [33], estimation of distribution [34], evolutionary, Bat [35], and symbiotic organism search algorithms [36] are among the sole meta-heuristic algorithms utilized for the solution of the RLP. Few research studies integrated various optimization methods with the meta-heuristic algorithms. Son and Skibniewski [37] combined a local optimizer with simulated annealing (SA). Doulabi et al. [38] proposed a GA with a local search heuristic for resource leveling with activity splitting. Alsayegh and Hariga [39] combined particle swarm optimization and SA methods to level resources while allowing activity splitting and considering splitting costs. Koulinas and Anagnostopoulos presented a simulated-annealing-based hyper-heuristic, and a tabu-search-based hyper-heuristic algorithm for leveling constrained resources [40]. Tabu-search-based methods were also implemented by Li et al. [41]. A non-dominated sorting genetic algorithm was implemented by Abadi [42]. A majority of the early meta-heuristic methods were validated by one or two case examples including up to 20 activities [5,21–24,29–32,37]. A few recent studies performed computational experiments for performance evaluation [25,38–42].

While the majority of the meta-heuristic research on resource leveling has focused on GAs, a sole GA may suffer from a rapid population convergence to local optima [43,44]. In contrast, SA has a fine-tuning capability and a good convergence property since its search is based on the cooling schedule, which specifies how the temperature is reduced as the search progresses [45]. However, a sole SA has a low search efficiency, as it maintains one solution at a time. In recent years, skilled combinations of GAs with SA were proposed to achieve an efficient search algorithm for many optimization problems [46–50].

In addition to the hybrid use of meta-heuristics, in recent years, the recognition of the limitations of sole optimization methods has led to the development of new optimization strategies by combining multiple methods to provide a more efficient behavior and higher flexibility when dealing with real-world and large-scale problems [51]. Memetic algorithms (MAs) were suggested within this context by hybridizing and combining existing algorithmic structures. MAs are extensions of evolutionary algorithms and are composed of an evolutionary framework and a local search algorithm. MAs are a pragmatic cross-disciplinary optimization paradigm and have been successfully applied in numerous fields including machine learning, knowledge discovery, economics, engineering, and scheduling [52].

Much of the research on the RLP has concentrated on designing heuristic and meta-heuristic solution methods [53]. The majority of these methods are limited to a single optimization strategy [5,21–25,29–32,37–42]. Piryonesi et al. [54] used a single metaheuristic strategy to solve the problem while activity splitting was allowed, Selvam and Tadepalli [55] used only a genetic algorithm, and Duraiswamy and Selvam [56] used ant colony optimization as a meta-heuristic method to solve the RLP problem. Damci et al. [57]

used the float consumption rate as an option during the leveling process, and Prayogo and Kusuma [58] used a multi-objective during the optimization process. A few studies included hybrid strategies [38]; however, very few published studies focused on incorporating the individual learning as a separate process for local refinement to design an efficient optimization strategy for the RLP. Integrating complementary strengths of different optimization methods and incorporating the individual learning as a separate process has a significant potential for achieving a successful optimization method for the RLP. Hence, the main focus of this paper is to narrow this gap in the literature by presenting a memetic algorithm (MA) that is composed of an evolutionary framework that includes a genetic algorithm (GA) with simulated annealing (SA) and a local search algorithm consisting of a shifting heuristic. The proposed algorithm was designed to achieve an effective optimization strategy for the RLP by integrating complementary strengths of different optimization methods and incorporating the individual learning as a separate process. The remainder of the paper is organized as follows: Section 2 is devoted to the novel memetic algorithm. In Section 3, two mixed-integer linear models are presented for the RLP to provide a basis for performance evaluation. An illustrative case study example is given in Section 4. The computational experiment results are presented in Section 5, and concluding remarks are made in Section 6.

2. Methodology of the Memetic Algorithm

A genetic algorithm is suitable for implementing multiple directional searches in a parallel architecture and can capture critical components of the past good solutions [59]. However, sole GAs often lack a sufficient search intensification capability. MAs were proposed to combine strengths of hierarchical population search methods with the intensification capabilities of local search procedures [60]. MAs offer a new problem-oriented algorithmic design perspective [52].

In this paper, an MA was designed specifically for the solution of the RLP. The evolutionary framework of the MA was developed based on a hybrid GA with SA. The GA enables searches in parallel architecture and captures critical components of past good solutions. The SA was used to control the search process for avoiding premature convergence [61]. The local search algorithm of the MA is composed of a shifting heuristic specific for the leveling problem. The shifting heuristic incorporates the individual learning for local refinement. The MA is described in the following subsections.

2.1. Chromosome Representation

In the evolutionary framework of the MA, candidate solutions to an optimization problem are represented by individuals. The solutions are encoded to an MA by chromosomes, which are a string of parameters called genes. The genes, composed of randomly generated real numbers between 0 and 1, represent the delay in non-critical activities, as shown in Equation (1).

$$delay\ time_i = rounddown\ (GV_i * [TF_i + 1]) \quad (1)$$

In Equation (1), GV_i is gene value between 0 and 1, and TF_i is the total float of the ith noncritical activity. The MA schedules the activities in the precedence-free activities list in ascending activity ID. The predecessor-free activity list consists of activities without any predecessors and the activities for which all of the predecessors have been scheduled. Higher priority for scheduling is assigned for the activity with a smaller ID. After determining the start and finish times of the activity with the highest priority, the next activity with the closest priority is examined. The procedure continues until all of the noncritical activities are scheduled.

2.2. Heuristic Improvement

The local search algorithm of the MA is composed of a shifting heuristic. The shifting heuristic attempts to improve the resource profile of a given schedule by searching the

delay time alternatives of non-critical activities one by one in ascending activity ID order without changing the start times of the remaining activities.

The heuristic search algorithm attempts to delay alternatives from 0 to TF_i. If the objective function is improved, the heuristic algorithm replaces the current solution with the obtained best solution. The gene representation of the improved solution is updated by Equation (2), in which the shift of the *i*th activity improves the objective function.

$$GV_i = \frac{delaytime_i + 0.5}{TF_i + 1} \tag{2}$$

The heuristic search finishes when all of the non-critical activities are investigated.

2.3. Crossover, Mutation, and Simulated Annealing

New individuals are introduced by crossover and mutation operators. The MA performs a two-point crossover. The mutation operator of the MA changes a gene value of a selected chromosome with a random real number between 0 and 1. SA is integrated with the evolutionary framework of the MA to perform mutations based on a cooling schedule. The MA executes a mutation that leads to an individual with a worse fitness evaluation function value if the condition in Equation (3) is true [61].

$$r \leq e^{\frac{(f - f')}{f} x \frac{B}{t}} \tag{3}$$

where r is a randomly generated real number between 0 and 1, f is the fitness value before mutation, f'' is the fitness value after the mutation, B is the Boltzmann constant, and t is the temperature. The main purpose of the simulated annealing integrated mutation strategy is to prevent premature convergence by controlling the search process more efficiently. In the initial search stages, mutations leading to a worse fitness value are allowed to avoid being trapped in certain solutions. In later stages, fewer mutations leading to a worse fitness value are allowed to achieve fine tuning and a good convergence property by decreasing the temperature based on a cooling schedule that specifies how the temperature is reduced as the search progresses. The MA is evolved toward better solutions via the elitist roulette wheel selection method. The flowchart of the MA is given in Figure 1.

2.4. Excel Interface

The MA was implemented using C# and compiled within Visual Studio 2010. The Microsoft Excel interface was integrated into the MA to simplify the problem input and to enable data exchange with the commercial project management software. The interface enabled integration of the MA with Primavera and Microsoft Project, which are commonly used for the planning and management of construction projects. The duration, precedence, and resource information of the project in Primavera or Microsoft Project can be exported as a Excel file and used as an input by the interface. Once leveling is performed by the interface, the optimal start times of the activities can be imported by Primavera or Microsoft Project.

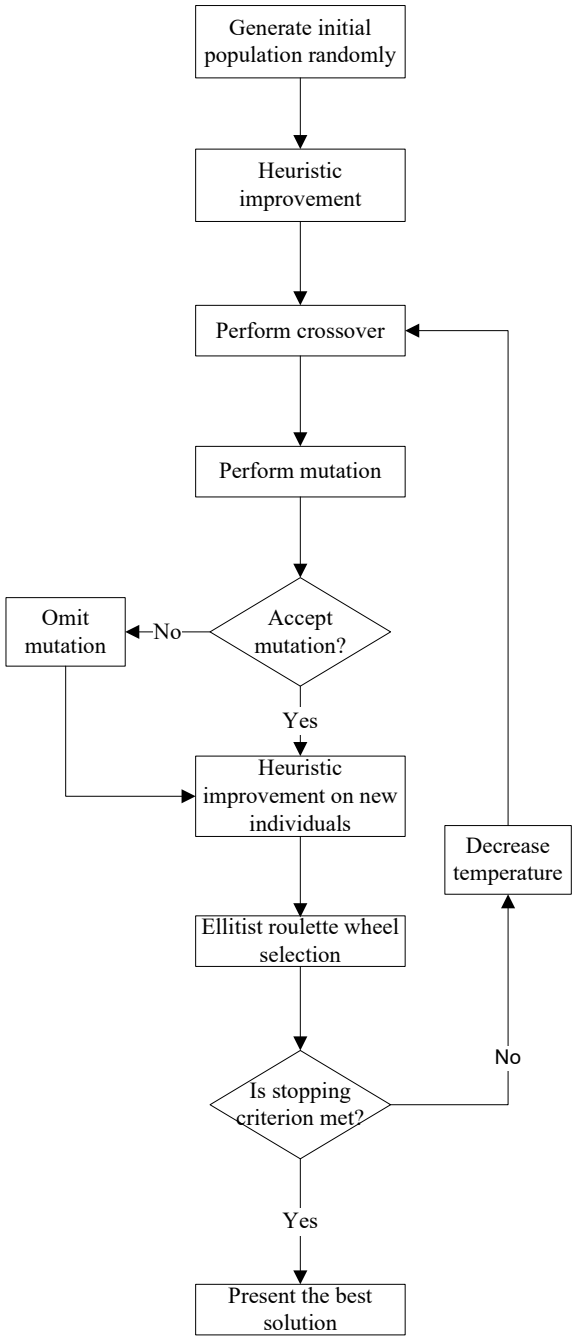

Figure 1. Flowchart of the MA.

3. Mixed-Integer Linear Models

In this study, two mixed-integer linear models are presented to obtain the optimum solutions for small instances. The models minimize the difference between fluctuating resource profiles and a predetermined desirable shape of a rectangular resource profile [24].

The optimum solutions are used in the performance evaluation of the MA. The first model considers the absolute deviation between the resource requirement and a targeted uniform resource level (ADIF) metric to minimize the fluctuations in the resource utilization curve, whereas the second model minimizes the SSRR. The ADIF and SSRR metrics are considered for the performance evaluation of MA since these metrics are the most commonly used leveling metrics in previous research. The models are an extension of the previous model presented for the combined resource idle days (RIDs) and maximum resource demand metric (MRD) [62]. However, the models presented for ADIF and SSRR

are significantly different than the RID-MRD model [62] since there are major differences between parameters, variables, and constraints.

3.1. Sets

Mixed-integer model is developed considering the below sets:

$$I = \{i_1, i_2, \ldots, i_i\} \text{ is the set of activities } I \tag{4}$$

$$T = \{t_1, t_2, \ldots, t_t\} \text{ is the set of time periods in which the activities can be scheduled} \tag{5}$$

t_1 is the beginning time and t_t is the completion time of the project.

$$R = \{r_1, r_2, \ldots, r_r\} \text{ is the set of resources} \tag{6}$$

$$N = \{n_1, n_2, \ldots, n_n\} \text{ is the set of total daily resource demands} \tag{7}$$

3.2. Parameters

$$EST_i = \text{Earliest start time of activity } i \tag{8}$$

$$LST_i = \text{Latest start time of activity } i \tag{9}$$

$$d_i = \text{Duration of activity } i \tag{10}$$

$$r_{i,r} = \text{Resource demand of activity } i \text{ for resource } r \tag{11}$$

$$a_{t,r} = \text{Targeted demand of resource } r \text{ for day } t \tag{12}$$

$$w_r = \text{Weight of resource } r \tag{13}$$

$$D = \text{Project duration} \tag{14}$$

$$\text{pi, j} : \begin{cases} 1, \text{if activity j should be finished before activity i} \\ 0, \text{otherwise} \end{cases} \tag{15}$$

3.3. Variables

$$z_1 = \text{The weighted sum of absolute deviations from the targeted resource demands} \tag{16}$$

$$z_2 = \text{The weighted sum of resource demands for all time periods} \tag{17}$$

$$f_i = \text{The start time of activity } i \tag{18}$$

$$u_{t,r} = \text{Resource demand at time period } t \text{ for resource } r \tag{19}$$

$$v_{t,r} = \text{Square of resource demand at time period } t \text{ for resource } r \tag{20}$$

$$x_{t,r} = \text{Amount of resource demand that is more than targeted demand } a_{t,r} \text{ at time period } t \text{ for resource } r \tag{21}$$

$$y_{t,r} = \text{Amount of resource demand that is less than targeted demand } a_{t,r} \text{ at time period } t \text{ for resource } r \tag{22}$$

$$\lambda_{\text{n,t,r}} : \begin{cases} 1, \text{if demand for resource r at time period t is equal to n} \\ 0, \text{otherwise} \end{cases} \tag{23}$$

$$\varphi_{t,i} : \begin{cases} 1, \text{if activity } i \text{ is under progress at time period } t \\ 0, \text{otherwise} \end{cases} \quad (24)$$

$$\sigma_{t,i} : \begin{cases} 1, \text{if activity } i \text{ has started at time period } t \\ 0, \text{otherwise} \end{cases} \quad (25)$$

3.4. Models

The first model, as shown in Equation (26), minimizes the absolute deviation between the resource requirement and a targeted uniform resource level (ADIF) [2]. The objective of the second model is to minimize the sum of squares of resource requirements (SSRR) for all time periods, as shown in Equation (27).

$$\min z_1 = \sum_t \sum_r w_r |u_{t,r} - a_{t,r}| \quad (26)$$

$$\min z_2 = \sum_t \sum_r w_r {u_{t,r}}^2 \quad (27)$$

Since both of the metrics are not linear, the metrics are expressed in terms of the linear models. The common constraints of the models are presented at the end of this section.

3.4.1. Model for ADIF

$$\min z_1 = \sum_t \sum_r w_r \left(x_{t,r} + y_{t,r}\right) \quad (28)$$

$$u_{t,r} - a_{t,r} = x_{t,r} - y_{t,r} \; \forall t \in T, \; \forall r \in R \quad (29)$$

$$x_{t,r}, y_{t,r} \in Z_0 \; \forall t \in T, \; \forall r \in R \quad (30)$$

The ADIF leveling metric is expressed as a linear objective function in Equation (28). The constraint given in Equation (29) expresses the $u_{t,r} - a_{t,r}$ term as difference of two non-negative integer variables, as the absolute value function is not linear.

3.4.2. Model for SSRR

$$\min z_2 = \sum_t \sum_r w_r v_{t,r} \quad (31)$$

$$u_{t,r} = \sum_n n \lambda_{n,t,r} \; \forall t \in T, \; \forall r \in R \quad (32)$$

$$v_{t,r} = \sum_n n^2 \lambda_{n,t,r} \; \forall t \in T, \; \forall r \in R \quad (33)$$

$$\sum_n \lambda_{n,t,r} = 1 \; \forall t \in T, \; \forall r \in R \quad (34)$$

$$v_{t,r} \in Z_0 \; \forall t \in T, \; \forall r \in R \quad (35)$$

$$\lambda_{n,t,r} \in \{0, 1\} \; \forall n \in N, \; \forall t \in T, \; \forall r \in R \quad (36)$$

The objective function given in Equation (31) minimizes the weighted SSRR for all time periods. Equation (32) determines the sum of resource requirements, and Equation (33) determines the SSRR for all time periods. Equation (34) ensures that the sum of resource requirements for resource r can take a unique value.

3.5. Common Scheduling Constraints

$$\sum_i r_{i,r} \varphi_{t,i} = u_{t,r} \; \forall t \in T, \; \forall r \in R \quad (37)$$

$$p_{i,j}f_i \geq p_{i,j}(f_j + d_j)\ \forall i,j \in I,\ i \neq j \tag{38}$$

$$\sum_{EST_i \leq t \leq LST_i} t\sigma_{t,i} = f_i\ \forall i \in I \tag{39}$$

$$\sum_{EST_i \leq t \leq LST_i} \sigma_{t,i} = 1\ \forall i \in I \tag{40}$$

$$\varphi_{t,i} = \sum_{t=\max(EST_i, t-d_i+1)}^{\min(LST_i,t)} \sigma_{t,i}\ \forall t \in T,\ \forall i \in I,\ EST_i \leq t \leq LST_i + d_i - 1 \tag{41}$$

$$\varphi_{t,i} = 0\ \forall t \in T,\ \forall i \in I,\ t < EST_i \tag{42}$$

$$\varphi_{t,i} = 0\ \forall t \in T,\ \forall i \in I,\ t > LST_i + d_i - 1 \tag{43}$$

$$f_1 = 0 \tag{44}$$

$$f_I \leq D \tag{45}$$

$$\sigma_{0,1} = 1 \tag{46}$$

$$u_{t,r} \in Z_0\ \forall t \in T,\ \forall r \in R \tag{47}$$

$$f_i \in Z_0\ \forall i \in I \tag{48}$$

$$\varphi_{t,i} \in \{0,1\}\ \forall t \in T,\ \forall i \in I \tag{49}$$

$$\sigma_{t,i} \in \{0,1\}\ \forall t \in T,\ \forall i \in I \tag{50}$$

The scheduling constraints are common in both models. Equation (37) determines the resource usage for time periods. Equation (38) satisfies the precedence constraints. Equation (39) determines the start times of the activities. Equation (40) ensures that an activity starts at a time between its early start and late start times. Equation (41) determines the time periods the activities are in progress. Equations (42) and (43) ensure that an activity cannot be executed outside the early start and late finish times. The first and the last activities are dummy activities that determine the project start time and project completion time. Equation (44) defines the start time of project as day 0. Equation (45) ensures that the project is not completed later than the finish time of all activities. Equation (46) starts the first activity at time 0.

4. Illustrative Case Study

The leveling example of Son and Skibniewski [37] is used to illustrate the chromosome representation along with the encoding and decoding scheme designed for the MA. In the case example, the sum of squares of resource requirements (SSRR) resource leveling metric is used to measure the fluctuations in the resource utilization curve. The case example includes six non-critical activities, as shown in Figure 2. An arbitrary chromosome representation for the example is given in Figure 3.

The initial precedence-free activity list includes activities 1 and 4. Activity-1 had a smaller activity ID, hence this activity was scheduled first. The duration (D) and resource requirement (RR) of Activity-1 were 8 days and two resources, respectively. In the initial schedule, the early start time (ES) of Activity-1 was day 0, and the late start time (LS) of this activity was day 7. The number of start time alternatives of Activity-1 was eight, as this activity had a total float (TF) of seven days. Eight intervals were formed between zero and one to determine the start time of Activity-1. Thus, the interval length was 0.125 (1/8), the randomly generated value of 0.240 corresponded to the second interval, and Activity-1 was

scheduled to start at the second start time alternative. Hence, the scheduled start time (SS) of Activity-1 was determined as day 1, and the scheduled finish time (SF) of Activity-1 was determined as day 9. Once Activity-1 was scheduled, the early start times, late start times, and total floats of all non-critical activities were updated. Then, Activity-1 was removed from the precedence-free activities list, and Activity-2 was added to the list. The procedure was continued with the next activity in the list until all of the activities were scheduled. The resulting schedule had an SSRR value of 985, as shown in Figure 4.

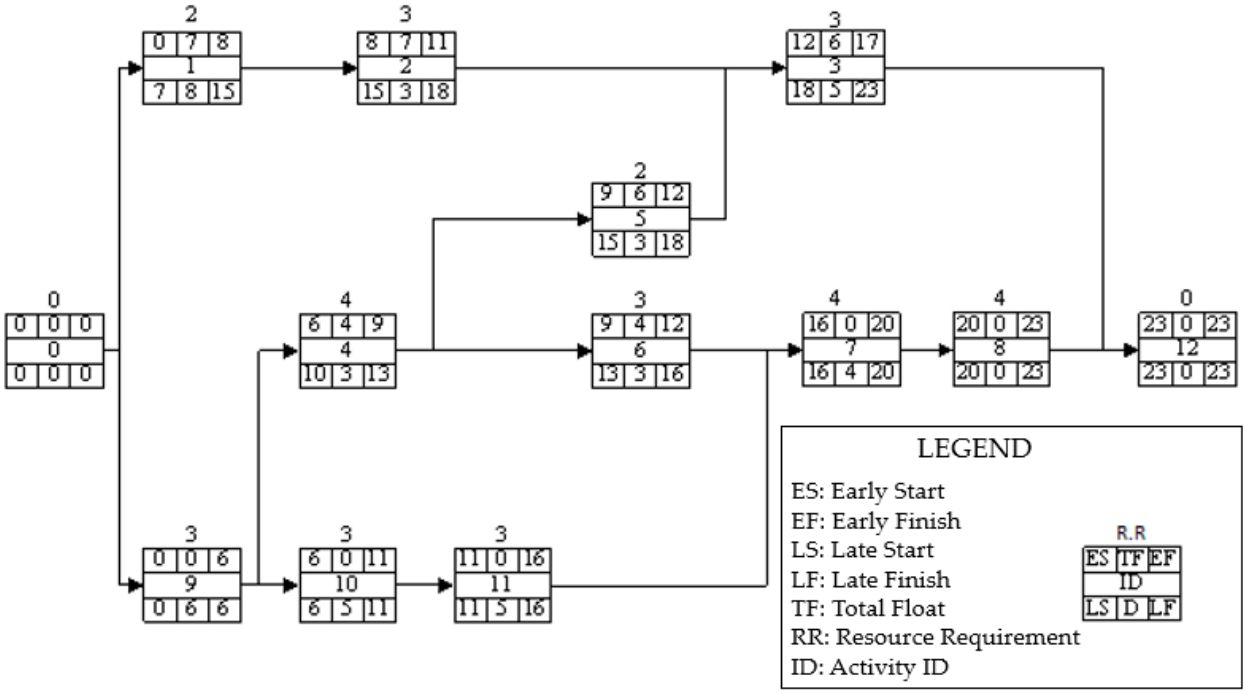

Figure 2. Case Example.

0.240	0.631	0.719	0.853	0.402	0.363

Figure 3. Chromosome representation for the example problem.

In order to illustrate the heuristic improvement, the shifting heuristic was applied to the schedule shown in Figure 4. Heuristic improvement for the first gene was explored first by evaluating the SSRR values of all possible start time alternatives for Activity-1 without changing the start times of the remaining activities. Starting Activity-1 at days 0, 2, 3, 4, and 5 were tried, and an improvement in the SSRR value could not be achieved. Hence, Activity-1 was not shifted. However, starting Activity-2 at day 15 instead of day 13 improved the SSRR value to 961. Therefore, the start time of Activity-2 was shifted to day 15, and the early start times, late start times, and total floats of all non-critical activities were updated. The procedure was applied to all remaining non-critical activities, and the SSRR value was improved to 957, as shown in Figure 5. The start times of the improved schedule were encoded by implementing Equation (2). To illustrate, Equation (2) provided 0.929 when Activity-2, which had 7 days of total float, was delayed 7 days. The chromosome representation of the improved schedule is given in Figure 6. The input screen of the interface for the case example is shown in Figure 7.

							Days																					
ID	ES	LS	SS	SF	D	RR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0	5	1	9	8	2		2	2	2	2	2	2	2	2														
2	9	15	13	16	3	3														3	3	3							
3	17	18	18	23	5	3																			3	3	3	3	3
4	6	10	10	13	3	4											4	4	4										
5	13	15	14	17	3	2															2	2	2						
6	13	13	13	16	3	3														3	3	3							
7	16	16	16	20	4	4																	4	4	4	4			
8	20	20	20	23	3	4																					4	4	4
9	0	0	0	6	6	3	3	3	3	3	3	3																	
10	6	6	6	11	5	3							3	3	3	3	3												
11	11	11	11	16	5	3												3	3	3	3	3							
Daily RR							3	5	5	5	5	5	5	5	5	3	7	7	7	9	11	11	6	4	7	7	7	7	7
Squared Daily RR							9	25	25	25	25	25	25	25	25	9	49	49	49	81	121	121	36	16	49	49	49	49	49

SSRR 985

LEGEND
ID: Activity ID
ES: Early Start
LS: Late Start
SS: Scheduled Start
SF: Scheduled Finish
RR: Resource Requirement
SSRR: Sum of squares of RR

Figure 4. Decoding of the initial chromosome representation.

							Days																					
	ES	LS	SS	SF	D	R	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0	7	1	9	8	2		2	2	2	2	2	2	2	2														
2	9	15	15	18	3	3																3	3	3					
3	18	18	18	23	5	3																			3	3	3	3	3
4	6	10	10	13	3	4											4	4	4										
5	13	15	13	16	3	2														2	2	2							
6	13	13	13	16	3	3														3	3	3							
7	16	16	16	20	4	4																	4	4	4	4			
8	20	20	20	23	3	4																					4	4	4
9	0	0	0	6	6	3	3	3	3	3	3	3																	
10	6	6	6	11	5	3							3	3	3	3	3												
11	11	11	11	16	5	3												3	3	3	3	3							
Daily RR							3	5	5	5	5	5	5	5	5	3	7	7	7	8	8	11	7	7	7	7	7	7	7
Squared Daily RR							9	25	25	25	25	25	25	25	25	9	49	49	49	64	64	121	49	49	49	49	49	49	49

SSRR 957

Figure 5. Decoding of the chromosome representation of the improved schedule.

0.188	0.929	0.500	0.900	0.167	0.500

Figure 6. Encoding of the improved schedule.

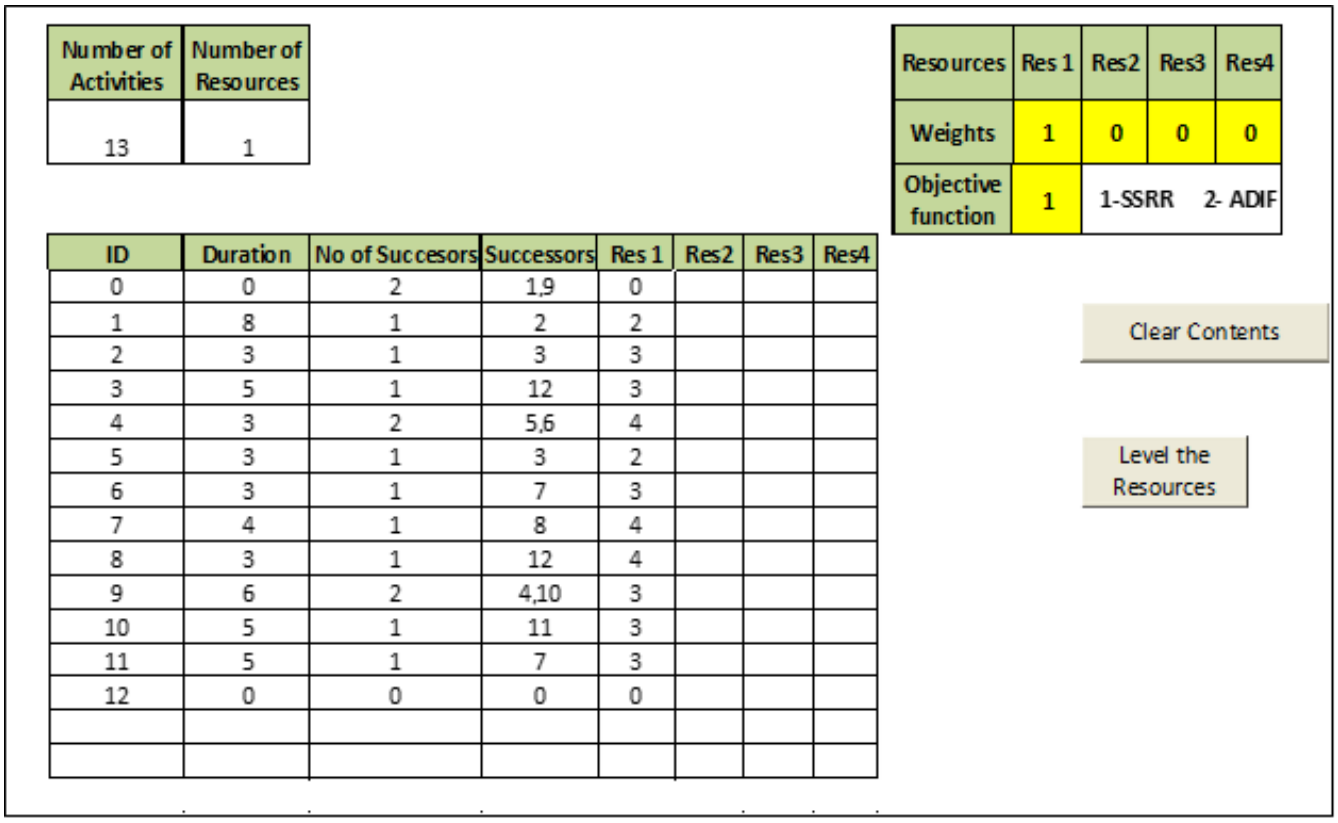

Figure 7. Excel interface.

5. Computational Experiments

In this section, the performance of the proposed MA is compared with the performance of the state-of-the-art heuristic and meta-heuristic methods. All of the tests were carried out on a computer with a 3.00 GHz Core 2 Duo Processor E8400 Intel CPU. A total of 1443 test instances including up to 120 activities and four resources were tested with two leveling metrics in the computational experiments. The parameters of the MA were configured based on a pilot study. In the computational experiments, a population size of 30, a crossover rate of 0.3, and a mutation rate of 0.2 are used. The percentage of mutations leading to a worse fitness value was decreased according to the simulated annealing procedure. The temperature was decreased based on a linear cooling scheme. The number of schedule evaluations was used as the stopping criterion. The pilot study revealed that the MA in general converged within 500,000 schedules. The convergence behavior of the MA for a 120-activity problem is illustrated in Figure 8. In the initial search stages, the MA improved the solutions rapidly, and at later stages, the MA executed fine tuning. In all computational experiments, the same parameters and 500,000 schedules were used to evaluate the performance of the MA.

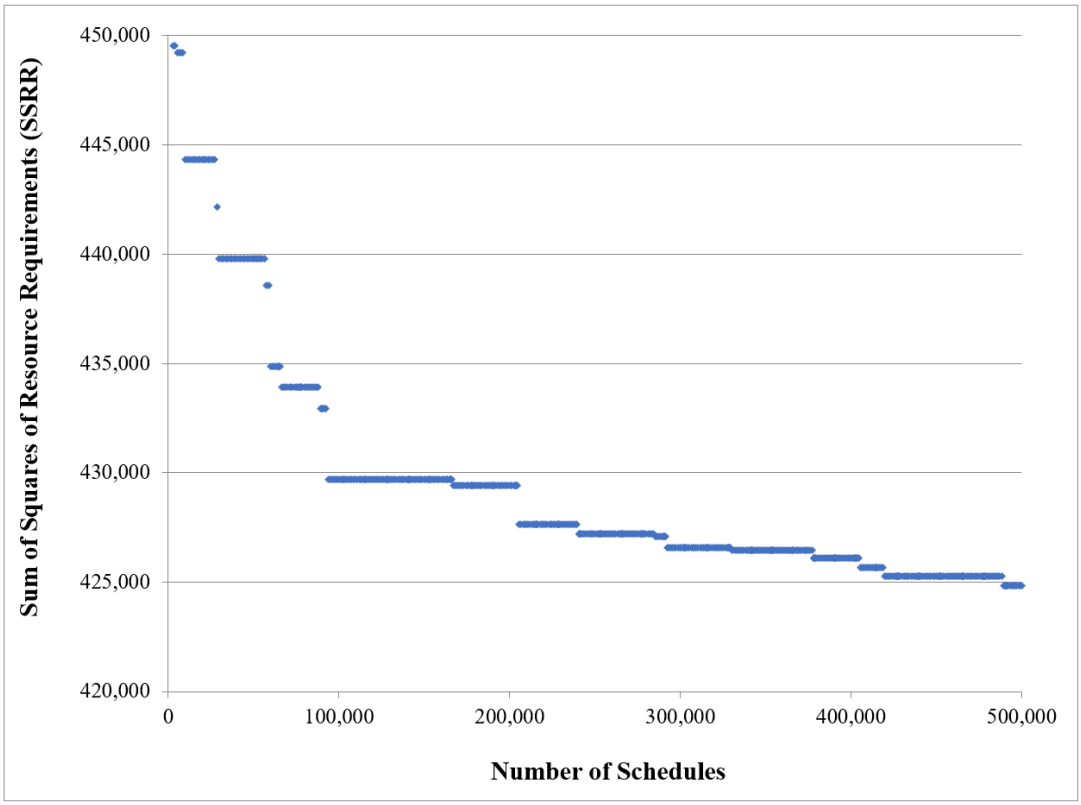

Figure 8. Convergence behavior of the MA.

5.1. Single Resource Case Examples

The majority of the leveling case examples in the literature included a single resource and a few activities. The performance of the MA was evaluated initially for the two case examples presented in Son and Skibniewski [37] and one case example presented in El-Rayes and Jun [24]. The stopping criterion for the MA was set as 500,000 schedule evaluations [63] for the single-resource case examples. Optimal solutions of the case examples were obtained by using the models presented in Equations (4)–(50). The targeted demand in Equation (12) was determined by rounding the average resource demand using the floor function. The results of the MA for the single-resource case examples are shown in Table 1. The MA was able to find the optimal result for all of the single case examples.

Table 1. Results of the MA for single-resource case examples.

Source	No. of Activities	Metric	Optimal	MA	Time (s)
Son and Skibniewski (1999) [37]	11	SSRR	915	915	3.4
Son and Skibniewski (1999) [37]	13	SSRR	6225	6225	1.9
El-Rayes and Jun (2009) [24]	20	SSRR	3059	3059	2.5
El-Rayes and Jun (2009) [24]	20	ADIF	90	90	2.5

5.2. Comparison of MA with Microsoft Project and Primavera

Primavera (Primavera Inc., Bala Cynwyd, PA,USA) and Microsoft Project (Microsoft Corp., Redmond, Washington, DC, USA) are the most commonly used software for the planning and management of construction projects [64]. Resource leveling can be performed in Primavera and Microsoft Project by setting targets for the resource demands. Despite the importance of leveling in practice, very few studies in the literature evaluated the performance of project management software for the RLP. In this section, the performance of the MA is compared first with the performance of nine priority-based leveling heuristics

available in Microsoft Project Professional 2013 and Primavera 6.7. The heuristics included the Standard (STD) heuristics of Microsoft Project (MSP) 2013 and the ID-Ascending (IDA), ID-Descending (IDD), Total Float-Ascending (TFA), Total Float-Descending (TFD), Early Start-Ascending (ESA), Early Start-Descending (ESD) Late Finish-Ascending (LFA), and Late Finish-Descending (LFD) heuristics of Primavera 6.7.

A total of 15 standard instances with 30 activities (J30), 15 standard instances with 60 activities (J60), and 15 standard instances with 120 activities (J120) were selected randomly from the project scheduling problem library (PSPLIB). The PSLIB instances were generated by utilizing a full factorial design of parameters of network complexity, resource factor, and resource strength, with 10 replications per cell. All problem instances included four resource types. Details of the test instances are described in Kolisch and Sprecher [65]. The ADIF leveling metric was used for comparison. The targeted demands for resources were determined by rounding the average resource demand for each resource using the floor function. The weights of all four resources were taken as equal in the computational experiments; however, the MA and the models presented in Section 3 could also solve leveling problems with different resource weights. All of the selected J30 test instances were solved to optimality within a computation time limit of five hours by using the standard solver CPLEX and the model presented for ADIF. However, optimal solutions were not obtained for the J60 and J120 instances within five hours. Hence, the average percentage deviation (APD) from the upper bounds (current best solutions) was used in comparisons for performance evaluation [65,66]. The stopping criterion for the MA was set as 500,000 schedule evaluations.

The performance comparison results are presented in Table 2. The APD of the MA from the optimal solutions was 0.5% for the J30 instances. The MA determined the best solution for 44 test instances among the 10 methods evaluated. The Total Float-Ascending heuristic obtained the best solution for the remaining instance. The average percentage deviations of the MA from the upper bounds were 0.0% and 0.1% for the J60 and J120 instances, respectively. The average CPU time for all the instances was 9.6 s. The MA produced very good results within a reasonable computing time.

Table 2. Comparison of the MA with Microsoft Project 2013 and Primavera 6.7.

Instance	**MSP**	**Primavera 6.7**								**This Study**	
Set	**STD**	**IDA**	**IDD**	**TFA**	**TFD**	**ESA**	**ESD**	**LFA**	**LFD**	**MA**	**T. (s)**
J30 (15)	47.5	42.9	48.9	48.0	43.6	45.0	41.8	45.9	44.4	0.5	4.7
J60 (15)	54.6	59.1	62.0	49.4	54.0	61.3	52.4	46.3	57.5	0.0	8.3
J120 (15)	51.9	54.4	50.4	38.1	57.9	66.2	53.0	45.0	59.7	0.1	15.8
Average	51.3	52.1	53.8	45.2	51.8	57.5	49.1	45.7	53.9	0.2	9.6

The nine priority based leveling heuristics performed very poorly in comparison to the MA. Among the tested nine heuristics, the Total Float-Ascending and Late Finish-Ascending methods performed relatively better. The average percentage deviations of these methods for all instances were 45.2% and 45.7% respectively, whereas the APD of MA for all instances was 0.2%. The performance gap between the MA and the nine priority-based leveling heuristics revealed the limitations of the commercial project management software for resource leveling.

5.3. Comparison of the MA with State-of-the-Art Heuristic and Meta-Heuristic Methods

In a recent study, Ponz-Tienda et al. [25] presented an adaptive GA (AGA) for RLP. Ponz-Tienda et al. [25] evaluated the performance of the AGA for the SSRR metric by using 480 J30 instances, 480 J60 instances, and 480 J120 instances. In Table 3, the performance of the MA is compared with the performance of the AGA. The modified version of the well-known Burgess shifting heuristic [17] is also included in the comparisons. The modified

Burgess algorithm (Burgess2) executes the standard Burgess method for several randomly selected activity ID lists and reports the best SSRR value achieved when the stopping criterion is met. The APD values given in Table 3 are the average percentage deviations from the current best solutions (upper bounds) for the SSRR metric. In the computational experiments, the equal weights of all four resources were taken. The J30 test instances were solved within the five-hour computation time limit by using the standard solver CPLEX and the model presented for the SSRR. Within the specified computation time limit, 475 J30 instances were solved to optimality. In the computational analysis, the result obtained by the MA were reported at the end of 500,000 schedule evaluations. The CPU time of the MA for each problem was used as the stopping criterion for the Burgess2 heuristic.

Table 3. Comparison of the MA with state-of-the-art heuristic and meta-heuristic methods.

	AGA [23]			Burgess2			MA (This Study)		
Instance	**APD**	**No. of**	**Time**	**APD**	**No. of**	**Time**	**APD**	**No. of**	**Time**
Set	**(%)**	**Optimal**	**(s)**	**(%)**	**Optimal**	**(s)**	**(%)**	**Optimal**	**(s)**
J30 (480)	0.7	76	15	3.6	14	12.6	0.2	232	12.6
J60 (480)	2.3	NA	NA	3.1	NA	18.3	0.0	NA	18.3
J120 (480)	3.7	NA	NA	2.1	NA	27.6	0.1	NA	27.6
Average:	2.2			2.9		19.5	0.1		19.5

Tables 3 and 4 present the summary of the computational results. The complete results for all instances and optimal solutions for J30 instances can be downloaded from https://docs.google.com/spreadsheet/ccc?key=0AvRxO1H9dRL6dFFrWHlJcTNhdl91TlNaNnBOS3RpckE&usp=sharing (accessed on 19 October 2023). The computational results of Table 3 indicate that the MA obtained better solutions that were either optimal or very close to optimal, as an APD of 0.2% was obtained for the J30 instances. Out of 475 J30 instances with optimal solutions, the MA was able to obtain the optimal for 232 instances. The AGA, with an APD of 0.7%, was the second-best method for the J30 instances and was able to determine the optimal for 76 instances. The computational experiments for the AGA were performed on a desktop computer with a 3.6 GHz Intel Core i7 processor. The average computing time of the AGA for the J30 instances was reported as 15 s [25]. For the J30 instances, the average CPU time of the MA on a desktop computer with a 3.00 GHz Intel E8400 Core 2 Duo Processor was 12.6 s. The MA was able to obtain better solutions compared to the AGA within a shorter computing time. Among the three methods evaluated, Burgess2 ranked last for the J30 instances and achieved an APD of 3.6%.

Table 4. Percentage improvement from the early start resource profile.

	AGA [23]		Burgess2		MA (This Study)	
Instance Set	**IES (%)**	**Time (s)**	**IES (%)**	**Time (s)**	**IES (%)**	**Time (s)**
J30 (480)	18.2	15	15.9	12.6	18.6	12.6
J60 (480)	23.1	NA	22.5	18.3	24.7	18.3
J120 (480)	27.9	NA	29.0	27.6	30.3	27.6
Average:	23.1		22.5	19.5	24.5	19.5

The MA obtained the best result for 467 J60 and 438 J120 instances, and it achieved an APD of 0.0% for the J60 and 0.1% for the J120 instances. The AGA had an APD of 2.3% for the J60 and 3.7% for the J120 instances. The APDs of Burgess2 for the J60 and J120 instances were 3.1% and 2.1%, respectively. The MA performed significantly better than the AGA and Burgess2 for all instance sets. Ponz-Tienda et al. [25] used percentage improvement from the early start resource profile (IES) to evaluate the performance of the AGA. The IES

values of the AGA, Burgess2, and the MA for the SSRR metric are presented in Table 4. The MA achieved the most average percentage improvement from the early start resource profile for the J30, J60, and J120 instances.

Performance measures of heuristics tend to cluster in three areas: solution quality, computational effort, and robustness [67]. The MA achieved high-quality solutions for most of the test instances, as the solutions of the MA were optimal or very close to the optimal for the J30 instances, and the solutions were the best solutions for majority of the J60 and J120 instances. The speed of computation is a key performance factor along with the quality of the solutions. The average CPU time of the MA for all instances was 19.5 s. The MA obtained high-quality solutions within a reasonable computation time. Robustness in general is based on the ability of a heuristic or meta-heuristic to perform well over a range of test problems [67].

6. Discussion

The computational experiment results indicate that the MA is a robust algorithm since it performed very well for the majority of the test instances with a different number of activities, network complexity, resource factor, and resource strength [65]. The MA is also a flexible algorithm since it can solve leveling problems with different leveling metrics such as ADIF or SSRR and with different resource weights. Since the MA's success was consistent for different problem sets with the same parameters, the parameter values presented can be generalizable to other problem instances. The computational experiment results confirmed the effectiveness of the MA.

Comparisons with Primavera and Microsoft Project revealed that the MA achieved significant improvements for the resource leveling capabilities of the existing project planning and management software. The Excel interface of the MA enabled data exchange with the commercial project management software. Hence, the MA enables a practical alternative for improving the resource leveling capabilities of project planning and management software.

7. Conclusions

This paper has multiple contributions. First, it presents a novel optimization strategy that combines complementary strengths of genetic algorithms, a shifting heuristic, and simulated annealing for the resource leveling problem (RLP) under an MA framework. Comparisons with popular commercial project management software and state-of-the-art methods validated the effectiveness of the proposed approach. Second, the paper reveals the limitations of the popular commercial project management software for resource leveling. Third, it presents mixed-integer linear models for two leveling metrics for solving the RLP to optimality. The optimal solutions obtained by the models provide a basis for performance evaluation.

The results reveal that the new optimization strategy of the MA achieved significant improvements for the RLP problem; hence, the objective of presenting a successful optimization method for RLP has been achieved. The pilot study and computational experiment results also indicate that the integration of an SA with a GA helps in improving the fine tuning and convergence property of GAs. The performance gap between the MA and the leveling heuristics of popular project management software revealed the potential for improving the heuristics of popular project management software for resource leveling. The MA provides an efficient leveling alternative that can be used along with the popular project management software for achieving optimal resource planning and management decisions.

The limitations of popular commercial project management software are also illustrated along with the improvements achieved by the MA to reveal potential contributions of the proposed integrated framework. Hence, the MA contributes to practice by enabling improvements for the resource leveling problem. The MA achieved good results for instances with up to 120 activities and four resources within a reasonable computing time. However, for larger problems, the computation time requirement of the MA may increase

significantly to achieve an adequate solution. To improve the effectiveness of the MA for projects including more activities and resources, integration of alternative optimization methods such as constraint programming or novel problem-specific heuristics appear to be promising areas for future research.

Author Contributions: Conceptualization, M.I. and R.S.; Methodology, M.I., R.S. and T.A.; Validation, M.I., T.A., F.U. and Ö.H.B.; Formal analysis, F.U.; Writing—original draft, M.I. and T.A.; Writing—review & editing, F.U. and Ö.H.B.; Supervision, R.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant #111M140.

Data Availability Statement: Not applicable.

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

References

1. Tavares, L.V. Optimal resource profiles for program scheduling. *Eur. J. Oper. Res.* **1987**, *29*, 83–90. [CrossRef]
2. Easa, S. Resource Leveling in Construction by Optimization. *J. Constr. Eng. Manag.* **1987**, *115*, 302–316. [CrossRef]
3. Ballestin, F.; Schwindt, C.; Zimmermann, J. Resource Leveling in Make-to-Order Production: Modelling and Heuristic Solution Method. *Int. J. Oper. Res.* **2007**, *4*, 50–62.
4. Son, J.; Mattila, K.G. Binary Resource Leveling Model: Activity Splitting Allowed. *J. Constr. Eng. Manag.* **2004**, *130*, 887–894. [CrossRef]
5. Iranagh, M.A.; Sonmez, R. A genetic algorithm for resource leveling of construction projects. In Proceedings of the 28th Annual ARCOM Conference, Edinburgh, UK, 3–5 September 2012; pp. 1047–1055.
6. Neumann, K.; Schwindt, C.; Zimmermann, J. *Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions*; Springer: Berlin/Heidelberg, Germany, 2002; Volume 508.
7. Ramlogan, R.; Goulter, I. Mixed integer model for resource allocation in project management. *Eng. Optim.* **1989**, *15*, 97–111. [CrossRef]
8. Mattila, K.G.; Abraham, D.M. Resource Leveling of Linear Schedules Using Integer Linear Programming. *J. Constr. Eng. Manag.* **1998**, *124*, 232–244. [CrossRef]
9. Hariga, M.; El-Sayegh, S. Cost optimization for the multi-resource leveling problem with allowed activity splitting. *J. Constr. Eng. Manag.* **2011**, *137*, 56–64. [CrossRef]
10. Rieck, J.; Zimmermann, J.; Gather, T. Mixed-integer linear programing for resource leveling problems. *Eur. J. Oper. Res.* **2012**, *221*, 27–37. [CrossRef]
11. Bianco, L.; Caramia, M.; Giordani, S. Resource levelling in project scheduling with generalized precedence relationships and variable execution intensities. *OR Spectr.* **2006**, *38*, 405–425. [CrossRef]
12. Tarasov, I.; Haït, A.; Battaïa, O. A Generalized MILP Formulation for the Period-Aggregated Resource Leveling Problem with Variable Job Duration. *Algorithms* **2020**, *13*, 6. [CrossRef]
13. Bandelloni, M.; Tucci, M.; Rinaldi, R. Optimal resource leveling using non-serial dynamic programming. *Eur. J. Oper. Res.* **1994**, *78*, 162–177. [CrossRef]
14. Mutlu, M.Ç. A Branch and Bound Algorithm for Resource Leveling Problem. Master's Thesis, Middle East Technical University, Ankara, Turkey, 2010.
15. Gather, T.; Zimmermann, J.; Bartels, J.H. Exact methods for the resource leveling problem. *J. Sched.* **2011**, *14*, 557–569. [CrossRef]
16. Bettemir, Ö.H.; Erzurum, T. Kaynak Dengeleme Probleminin Arama Uzayını Paralel Programlama ile Tarayarak Kesin Çözümü. *Tek. Dergi* **2021**, *32*, 10767–10805. [CrossRef]
17. Burgess, A.R.; Killebrew, J.B. Variation in Activity Level on a Cyclic Arrow Diagram. *J. Ind. Eng.* **1962**, *13*, 76–83.
18. He, L.; Zhang, L. Dynamic priority rule-based forward-backward heuristic algorithm for resource levelling problem in construction project. *J. Oper. Res. Soc.* **2013**, *64*, 1106–1117. [CrossRef]
19. Harris, R.B. Packing Method for Resource Leveling (PACK). *J. Constr. Eng. Manag.* **1990**, *116*, 331–350. [CrossRef]
20. Hiyassat, M.A.S. Applying Modified Minimum Moment Method to Multiple Resource Leveling. *J. Constr. Eng. Manag.* **2001**, *127*, 192–198. [CrossRef]
21. Chan, W.T.; Chua, D.K.H.; Kannan, G. Construction Resource Scheduling with Genetic Algorithms. *J. Constr. Eng. Manag.* **1996**, *122*, 125–132. [CrossRef]
22. Hegazy, T. Optimization of Resource Allocation and Leveling Using Genetic Algorithms. *J. Constr. Eng. Manag.* **1999**, *125*, 167–175. [CrossRef]
23. Leu, S.; Yang, C.; Huang, J. Resource Leveling in Construction by Genetic Algorithm-Based Optimization and Its Decision Support System Application. *Autom. Constr.* **2000**, *10*, 27–41. [CrossRef]

24. El-Rayes, K.; Jun, D.H. Optimizing Resource Leveling in Construction Projects. *J. Constr. Eng. Manag.* **2009**, *135*, 1172–1180. [CrossRef]
25. Ponz-Tienda, J.L.; Yepes, V.; Pellincer, E.; Moreno-Flores, J. The Resource Leveling Problem with multiple resource using adaptive genetic algorithm. *Autom. Constr.* **2013**, *29*, 161–172. [CrossRef]
26. Li, H.; Demeulemeester, E. A genetic algorithm for the robust resource leveling problem. *J. Sched.* **2016**, *19*, 43–60. [CrossRef]
27. Li, H.; Xiong, L.; Liu, Y.; Li, H. An effective genetic algorithm for the resource levelling problem with generalised precedence relations. *Int. J. Prod. Res.* **2018**, *56*, 2054–2075. [CrossRef]
28. Li, H.; Zheng, L.; Zhu, H. Resource leveling in projects with flexible structures. *Ann. Oper. Res.* **2023**, *321*, 311–342. [CrossRef]
29. Savin, D.; Alkass, S.; Fazio, P. A procedure for calculating the weight-matrix of a neural network for resource leveling. *Adv. Eng. Softw.* **1997**, *28*, 271–283. [CrossRef]
30. Kartam, N.; Tongthong, T. An artificial neural network for resource leveling problems. *Artif. Intell. Eng. Des. Anal. Manuf.* **1998**, *12*, 273–287. [CrossRef]
31. Qi, J.X.; Wang, Q.; Guo, X.Z. Improved particle swarm optimization for resource leveling problem. In Proceedings of the 6th International Conference on Machine Learning and Cybernetics, Hong Kong, China, 19–22 August 2007; pp. 896–901.
32. Geng, J.Q.; Weng, L.P.; Liu, S.H. An improved ant colony optimization algorithm for nonlinear resource leveling problems. *Comput. Math. Appl.* **2011**, *61*, 2300–2305. [CrossRef]
33. Xu, X.; Hao, J.; Zheng, Y. Multi-objective artificial bee colony algorithm for multi-stage resource leveling problem in sharing logistics network. *Comput. Ind. Eng.* **2020**, *142*, 106338. [CrossRef]
34. Li, H.; Dong, X. Multi-mode resource leveling in projects with mode-dependent generalized precedence relations. *Expert Syst. Appl.* **2018**, *97*, 193–204. [CrossRef]
35. Li, H.; Wang, M.; Dong, X. Resource leveling in projects with stochastic minimum time lags. *J. Constr. Eng. Manag.* **2019**, *145*, 04019015. [CrossRef]
36. Prayogo, D.; Cheng, M.Y.; Wong, F.T.; Tjandra, D.; Tran, D.H. Optimization model for construction project resource leveling using a novel modified symbiotic organisms search. *Asian J. Civ. Eng.* **2018**, *19*, 625–638. [CrossRef]
37. Son, J.; Skibniewski, M.J. Multiheuristic Approach for Resource Leveling Problem in Construction Engineering: Hybrid Approach. *J. Constr. Eng. Manag.* **1999**, *125*, 23–31. [CrossRef]
38. Doulabi, S.H.H.; Seifi, A.; Shariat, S.Y. Efficient Hybrid Genetic Algorithm for Resource Leveling via Activity Splitting. *J. Constr. Eng. Manag.* **2011**, *137*, 137–146. [CrossRef]
39. Alsayegh, H.; Hariga, M. Hybrid meta-heuristic methods for the multi-resource leveling problem with activity splitting. *Autom. Constr.* **2012**, *27*, 89–98. [CrossRef]
40. Koulinas, G.K.; Anagnostopoulos, K.P. A new tabu search-based hyper-heuristic algorithm for solving construction levelling problems with limited resource availabilities. *Autom. Constr.* **2013**, *31*, 169–175. [CrossRef]
41. Xu, H.Z.; Demeulemeester, E. Scheduling policies for the stochastic resource leveling problem. *J. Constr. Eng. Manag.* **2015**, *141*, 04014072.
42. Abadi, N.S.; Bagheri, N.; Assadi, M. Multiobjective model for solving resource-leveling problem with discounted cash flows. *Int. Trans. Oper. Res.* **2018**, *25*, 2009–2030. [CrossRef]
43. Rudolph, G. Convergence properties of canonical genetic algorithms. *IEEE Trans. Neural Netw.* **1994**, *5*, 96–101. [CrossRef]
44. Leung, Y.; Gao, Y.; Xu, Z.B. Degree of population diversity-a perspective on premature convergence in genetic algorithms and its Markov-chain analysis. *IEEE Trans. Neural Netw.* **1997**, *8*, 1165–1176. [CrossRef]
45. Hajek, B. Cooling schedules for optimal annealing. *Math. Oper. Res.* **1988**, *13*, 311–329. [CrossRef]
46. Hwang, S.F.; He, R.S. Improving real-parameter genetic algorithm with simulated annealing for engineering problems. *Adv. Eng. Softw.* **2006**, *37*, 406–418. [CrossRef]
47. Wang, L.; Zheng, D.Z. An effective hybrid optimization strategy for job-shop scheduling problems. *Comput. Oper. Res.* **2001**, *28*, 585–596. [CrossRef]
48. Sonmez, R.; Bettemir, O.H. A hybrid genetic algorithm for the discrete time–cost trade-off problem. *Expert Syst. Appl.* **2012**, *39*, 11428–11434. [CrossRef]
49. Chen, P.H.; Shahandashti, S.M. Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. *Autom. Constr.* **2009**, *18*, 434–443. [CrossRef]
50. Bettemir, Ö.H.; Sonmez, R. Hybrid genetic algorithm with simulated annealing for resource-constrained project scheduling. *J. Manag. Eng.* **2015**, *31*, 04014082. [CrossRef]
51. Blum, C.; Roli, A. Hybrid Metaheuristics: An Introduction. *Stud. Comput. Intell.* **2015**, *114*, 1–30.
52. Neri, F.; Cotta, C.; Moscato, P. (Eds.) *Handbook of Memetic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2012.
53. 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]
54. Piryonesi, S.M.; Nasseri, M.; Ramezani, A. Resource leveling in construction projects with activity splitting and resource constraints: A simulated annealing optimization. *Can. J. Civ. Eng.* **2019**, *46*, 81–86. [CrossRef]
55. Selvam, G.; Tadepalli, T.C. Genetic algorithm based optimization for resource leveling problem with precedence constrained scheduling. *Int. J. Constr. Manag.* **2019**, 1–10. [CrossRef]

56. Duraiswamy, A.; Selvam, G. An Ant Colony-Based Optimization Model for Resource-Leveling Problem. In *Advances in Construction Management: Select Proceedings of the ACMM 2021, Chengdu, China, 20–24 October 2021*; Springer: Singapore, 2022; pp. 333–342.
57. Damci, A.; Polat, G.; Akin, F.D.; Turkoglu, H. Use of float consumption rate in resource leveling of construction projects. *Front. Eng. Manag.* **2022**, *9*, 135–147. [CrossRef]
58. Prayogo, D.; Kusuma, C.T. Optimization of resource leveling problem under multiple objective criteria using a symbiotic organisms search. *Civ. Eng. Dimens.* **2019**, *21*, 43–51. [CrossRef]
59. Holland, J.H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, USA, 1975.
60. Moscato, P.; Norman, M.G. A Memetic approach for the traveling salesman problem implementation of a computational ecology for combi-combinatorial optimization on message-passing systems. *Parallel Comput. Transp. Appl.* **1992**, 177–186.
61. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
62. Iranagh, M.A.; Atan, T.; Sonmez, R. A mixed-integer linear model for optimization of resource idle days in project scheduling. In Proceedings of the Creative Construction Conference, Budapest, Hungary, 6–9 July 2013; pp. 368–381.
63. Kolisch, R.; Hartmann, S. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *Eur. J. Oper. Res.* **2006**, *174*, 23–37. [CrossRef]
64. Liberatore, M.J.; Pollack-Johnson, B.; Smith, C.A. Project management in construction: Software use and research directions. *J. Constr. Eng. Manag.* **2001**, *127*, 101–107. [CrossRef]
65. Kolisch, R.; Sprecher, A. PSPLIB—A Project Scheduling Library. *Eur. J. Oper. Res.* **1997**, *96*, 205–216. [CrossRef]
66. Debels, D.; Vanhoucke, M. A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Oper. Res.* **2007**, *55*, 457–469. [CrossRef]
67. Barr, R.S.; Golden, B.L.; Kelly, J.P.; Resende, M.G.; Stewart, W.R. Designing and reporting on computational experiments with heuristic methods. *J. Heuristics* **1995**, *1*, 9–32. [CrossRef]