Lagrange's Interpolation Embedded Multi-objective Genetic Algorithm to Solve Non-linear Multi-objective Optimization Problems

Muskan Kapoor[1], Bhupendra Kumar Pathak[1,*] and Rajiv Kumar[2]
[1]Department of Mathematics, Jaypee University of Information Technology, Solan, HP, India
[2]Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Solan, HP, India
E-mail: mkapoor5628@gmail.com, pathak.maths@gmail.com, rjv.ece@gmail.com
*Corresponding author

Student paper

Keywords: Lagrange interpolation, multi-objective optimization problems, multi-objective genetic algorithm

Received: May 26, 2023

This study describes a novel strategy for solving non-linear multi-objective optimization problems encountered in real-world engineering projects. To find the best trade-off points, a Lagrange's Interpolation embedded multi-objective genetic algorithm (LI-MOGA) is used. In this approach, Lagrange's Interpolation (LI) method is used to capture the non-linear relationship between time and cost. After that, LI is combined with MOGA to create a comprehensive strategy for solving non-linear multi-objective optimization problems in the real world. The study has implications for real-time monitoring and control of the project scheduling process.

Povzetek: Predstavljen je nov pristop za reševanje nelinearnih večciljnih optimizacijskih problemov z uporabo Lagrangeove interpolacije in večciljnega genetskega algoritma (LI-MOGA), ki optimizira čas in stroške projektov.

1 Introduction

Multi-Objective Genetic Algorithm (MOGA) is an optimization technique that utilizes genetic algorithms [1] to solve problems with multiple conflicting objectives. In contrast to traditional single-objective optimization problems, MOGA considers multiple objectives simultaneously and aims to find the best possible trade-offs between them. Typically, there are multiple solutions available rather than just one, and all of these solutions are Pareto optimal [3]. MOGA is based on the principles of natural selection and genetic recombination [1], [2]. It works by evolving a population of candidate solutions to the problem, which are represented as chromosomes in a genetic algorithm. These chromosomes are evaluated based on multiple objectives, rather than a single objective, and a fitness score is assigned to each chromosome based on its performance across all objectives. The goal of MOGA is to find a set of solutions that are Pareto-optimal, meaning that no other solution in the search space is better in all objectives simultaneously. This set of solutions is known as the Pareto Front, and the process of searching for this front is known as Pareto optimization. MOGA has applications in a wide range of fields, including engineering, finance, and biology. It is particularly useful in situations where multiple objectives need to be optimized simultaneously, and where there are trade-offs between these objectives that need to be considered. Agdas et al. [4] demonstrate that the meta-heuristic approach is efficient for large-scale constructing TCT problems and that GA can be utilized to solve vast benchmark networks of variables with high levels of accuracy consistently. Genetic algorithms (GA) and linear programming (LP) are combined to create a hybrid GALP method, which is used to address optimization problems. The outcomes of the proposed model are described in [5]. Recent developments in the genetic algorithm were examined by Katoch et al. [7]. In their research paper, they examine the potential possibilities for future study in the fields of genetic operators, fitness functions, and hybrid algorithms. The competitive environment makes the relationship between time and cost even more significant, as demonstrated by Albayrak [8]. The two project components, which conflict with one another and are affected by various project constraints, must be balanced. They presented a novel hybrid algorithm (NHA) as a solution to the TCT problem, which is viewed in their paper as a multi-objective optimization problem. Likewise, several Multi-Objective Evolutionary Algorithms (MOEAs) have been suggested to find the solutions of multi-objective optimization problems. The goal of Chassiakos and Rampis' research, [9] is to examine and assess the performance potential of various evolutionary algorithms described in [9] in order to address the time-cost trade-off problems. Venkatesh et al. [10] offered a new variation of the genetic algorithm designed to solve multi-

variable, multi-objective, and very high search space optimization problems. Their outcomes demonstrate that this algorithm is able to deliver positive and exact outcomes. Since it is given in real-time applications, [11] the issue of solving linear and nonlinear systems of equations is significantly beneficial and of great importance. Hassan et al. [11] introduced a comparison of several numerical approaches and GAs for solving an equation system. To distinguish between the best and worse options in order to get the best solutions, the use of genetic algorithms in numerous fields is demonstrated by Haldurai et al. [12] in their study along with how GA may be integrated with a number of other approaches to arrive at an optimal solution. Sharma et al. [13] provided a genetic algorithm-based TCT model, which was developed in a way that makes it simpler to identify the best techniques for finishing the project on schedule and for the least amount of cost. Pathak et al. [14] explained in their study how project uncertainties affect the non-linear time-cost trade-off (TCT) profile of actual engineering projects. Their study helps project managers come up with the optimal strategy for finishing a project under ambiguous conditions while also optimizing both costs and time. Most of the researchers used the evolutionary multi-objective optimization algorithms to solve linear multi-objective optimization problems like time-cost trade-off problems. In these methods, genetic operators are crucial for locating the real Pareto front.

But in non-linear MOOP, LI with evolutionary multi-objective optimization algorithms has not been investigated by researchers. This study describes an innovative and reliable approach, Lagrange's Interpolation embedded multi-objective genetic algorithm (LI-MOGA), that incorporates LI and MOGA. In this approach, LI method [15] is used to capture the non-linear relationship between time and cost. Thereafter, LI is integrated with MOGA to create an exhaustive method for solving non-linear MOOP in the real world.

2 Problem description

The non-linear relationship between project time and cost of an activity is shown in Figure 1 [6]. Suppose that a project network consists of n activities. Each activity i can be performed with (α_i) options with a corresponding cost c_i and time duration t_i. Let us assume that the vector of decision variables is taken as $X = [x_1, x_2, ..., x_i, ..., x_n]$. Now, the multi-objective TCT problem is defined by Eq. (1) and Eq. (2), respectively:

$$\text{minimize time } T = \sum_{i=1}^{n} t_i \tag{1}$$

$$\text{minimize cost } C = \sum_{i=1}^{n} c_i \tag{2}$$

where t_i and c_i can take any (α_i) options of i^{th} activity.

The project duration T is determined by computing the maximum path time (critical path method).

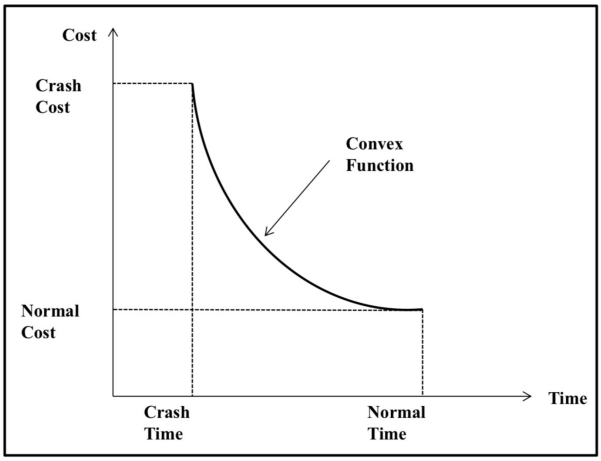

Figure 1: Non-linear time-cost trade-off relationship for an activity

3 Modelling of time-cost relationship with LI

Let's assume we have a set of input-output pairs, where the input represents the estimated completion time for a project, and the output represents the estimated cost. We can represent this set of pairs as $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$.

Using Lagrange interpolation, [17], [18] we can estimate the cost C for any given completion time T within the range of the collected input-output pairs. The Lagrange interpolation formula is:

$$\begin{aligned} f(x) = {} & \frac{(x-x_1)(x-x_2)(x-x_3)...(x-x_n)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)...(x_0-x_n)} y_0 \\ & + \frac{(x-x_0)(x-x_2)(x-x_3)...(x-x_n)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)...(x_1-x_n)} y_1 + ... \\ & + \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)...(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)(x_n-x_3)...(x_n-x_{n-1})} y_n \end{aligned} \tag{3}$$

To optimize the time and cost of the project simultaneously, we can use LI-MOGA to search for the optimal project completion time that minimizes cost while meeting the required project deadline. LI-MOGA generates a population of project activity completion times and applies the Lagrange interpolation formula to estimate the corresponding cost for each activity based on the project completion time.

4 Preliminaries of LI-MOGA

The following are the preliminary concepts of the LI-MOGA scheme for the MOOP, as used in this study:

4.1 Initial population

The initial population in LI-MOGA is typically generated randomly within the feasible search space, subject to any constraints that may be present in the problem. The initial population serves as the starting point for the optimization process and is evolved over multiple generations using genetic operators. The mathematical formulation of an initial population generated by LI-MOGA can be represented as follows:
The initial population consists of n_p solutions, where the solution is a string of type $[t_1, t_2, ..., t_i, ..., t_n]$ (Figure 2) such that $CT_i \leq t_i \leq NT_i$ and $i = 1, 2, ...n$, where CT_i and NT_i is the respective crash time and normal time of each activity.

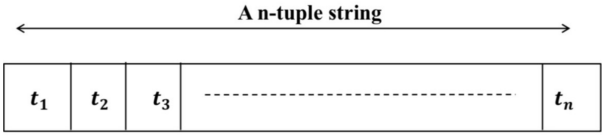

Figure 2: Time of the project schedule

The shortest time that may be used to complete an activity for a project is known as the crash time, and the expense that goes along with it is known as the crash cost and vice-versa. Each string's associated with project duration (T) and project costs (C) are determined by calculating the maximum path time and adding the costs for each activity, which is provided by LI respectively. These solutions are known as 'parents'.

4.2 Trade-off points and convex hull

In LI-MOGA, trade-off points and the convex hull of the Pareto Front are important concepts used to analyze the trade-offs between different objectives and identify the best solutions for a given problem. Trade-off points refer to the points on the Pareto Front that represent the optimal trade-offs between two objectives. Mathematically, for two objectives f_1 and f_2, the trade-off points can be defined as follows:
For a given solution x_i, let $f_i(x_i)$ be the value of the ith objective function. Then, the trade-off point $t(f_1, f_2)$ between objectives f_1 and f_2 can be defined as: $t(f_1, f_2) = (f_1(x_i), f_2(x_i))$ where x_i is a non-dominated solution that lies on the Pareto Front.

The convex hull of the Pareto front is a geometric shape that represents the set of all possible trade-offs between the objectives. The convex hull of the Pareto Front can be defined as follows:
Let N be the set of non-dominated solutions on the Pareto Front. Then, the convex hull of N can be represented as a set $CH(N)$, which contains all the extreme solutions and all the trade-off points between the objectives. Also, the extreme solution is a solution that is not dominated by any other solution in N.

4.3 Distance vs. fitness measure

The smallest distance (d_i) between the parent and each convex hull segment is determined as

$$f_i = d_{max} - d_i \tag{4}$$

$$p_i = \frac{f_i}{\sum f_i} \tag{5}$$

where p_i = probability of choosing parent i; f_i = fitness value of parent i; d_i = minimal distance between parent i and each segment j of CH; and d_i = min (d_{ij}, for all j).

4.4 Crossover operator used in LI-MOGA

Crossover is one of the important genetic operators [7] used in multi-objective genetic algorithms to generate new candidate solutions from parent solutions. The crossover operator involves exchanging genetic information between two parent solutions to create a new offspring solution.
The mathematical formulation of crossover can be represented as follows:
Let P be the population of candidate solutions, and let x_i and x_j be two parent solutions selected from P for crossover. Let n be the number of decision variables in each solution. The crossover operator involves randomly selecting a crossover point k, whereas k lies between $1 \leq k \leq n$ and exchanging the genetic information between the parents at this point. First k positions of child are taken from first k positions of x_i while remaining $(n-k)$ positions are defined by $(n-k)$ positions of x_j.
The offspring solutions are then evaluated based on their fitness using the fitness function. The offspring solutions can also undergo mutation, another genetic operator, to introduce further genetic diversity into the population as defined in the subsection 4.5.

4.5 Mutation opertor used in LI-MOGA

Mutation is a genetic operator [7] used in multi-objective genetic algorithms to introduce new genetic material into the population of candidate solutions. The mutation operator involves randomly changing the value of a decision variable in a candidate solution to create a new offspring solution.

Let P represent the population of candidate solutions, and let $x_i \in P$ be a parent solution selected for mutation, where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]$ is a vector of n decision

variables. The mutation operator involves randomly selecting a decision variable j ($1 \leq j \leq n$) and changing its value to a new value x'_{ij} within the feasible search space. The resulting offspring solution y_i can be expressed as:

$$y_i = [x_{i1}, x_{i2}, \ldots, x_{i(j-1)}, x'_{ij}, x_{i(j+1)}, \ldots, x_{in}] \quad (6)$$

Here, x_{ij} is the original value of the j-th decision variable in the parent solution x_i, and x'_{ij} is the new value assigned to the j-th decision variable, generated within its allowable range. After mutation, the offspring solution y_i is evaluated using the fitness function. If the new solution improves the population's diversity or satisfies specific selection criteria, it may replace an existing solution in P to maintain a constant population size.

5 Working methodology of LI-MOGA

The Lagrange interpolation method is a mathematical technique for constructing a polynomial that passes through a given set of data points. Multi-objective genetic algorithm are a class of optimization algorithms that use genetic operators such as selection, crossover, and mutation to search for solutions that optimize multiple objectives simultaneously. The LI-MOGA combines these two methods to solve optimization problems.

Here, LI-MOGA scheme can be summarized as follows:

1. Define the problem: Identify the problem to be solved and the objective functions to be optimized.
2. Generate the initial population: Create an initial population of candidate solutions using random values within the problem's constraints as mentioned in subsection 4.1.
3. Select the sets of individuals i.e. non-dominating solutions known as trade-off points and draw the convex hull as given in subsection 4.2.
4. Calculate fitness: Evaluate the fitness of each candidate solution as detailed out in subsection 4.3.
5. Select parents: Use the tournament selection method to choose parent solutions for the next generation.
6. Crossover: Perform crossover on the selected parents to create new candidate solutions, refer subsection 4.4.
7. Mutation: Apply mutation to the new candidate solutions to introduce additional diversity in the population as mentioned in subsection 4.5.
8. Replace the intial population with new population.
9. Repeat: Repeat steps 3-8 until a stopping criterion is met, such as reaching a maximum number of generations or achieving a satisfactory level of convergence.

6 Benchmark problems using MOGA

Using MOGA, two common test problems Zitzler-Deb-Thiele's (ZDT1 and ZDT2) with convex Pareto-optimal front are successfully attempted [16]. It is evident from visualising the findings (as shown in graphs) that MOGA generates non-dominating solutions which are acceptable from the perspectives of diversity and convergence.

6.1 ZDT1 benchmark problem

The ZDT1 benchmark problem has two objectives which have to be minimized as illustrated below:

$$ZDT1: \begin{cases} f_1(v) & = v_1 \\ f_2(v) & = h(v)[1 - \sqrt{\frac{v_1}{h(v)}}] \\ h(v) & = 1 + \frac{9}{n-1}\sum_{i=2}^{n} v_i \end{cases}$$

This problem [16] has 30 design variables which lie in the range [0, 1] and convex Pareto-optimal solutions lies in the range $0 \leq v_1^* \leq 1$ and $v_i^* = 0$ for $i = 2, 3, ..., 30$. Ac-

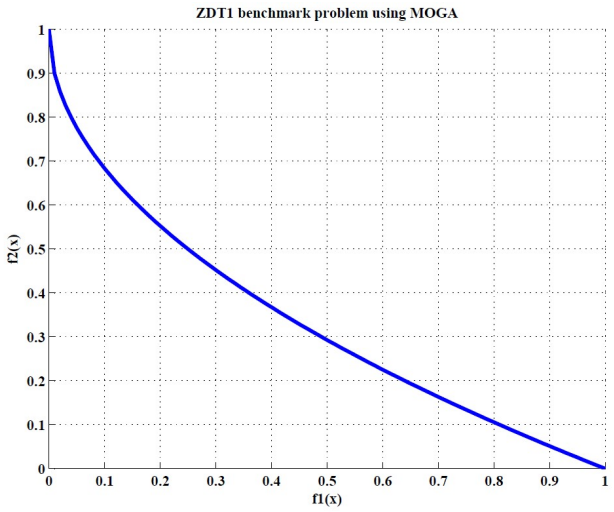

Figure 3: ZDT1 benchmark problem using MOGA

cording to Figure 3, the non-dominated solutions obtained using the MOGA technique are well dispersed throughout the solution space and matches the Pareto-optimal front in a reasonable way. Thus, the MOGA technique proves its capacity for convergence to the Pareto-optimal front and for discovering a variety of solutions for complex situations.

6.2 ZDT2 benchmark problem

The ZDT2 function has a non-convex Pareto-optimal front. The objective functionas are as mentioned below;

$$ZDT2: \begin{cases} f_1(v) & = v_1 \\ f_2(v) & = h(v)[1 - (\frac{v_1}{h(v)})^2] \\ h(v) & = 1 + \frac{9(\sum_{i=2}^{n} v_i)}{n-1} \end{cases}$$

Thirty design variables v_i ; $i = 1, 2, ..., n$ were selected for this ZDT2 function $(n = 30)$. Every design variable had a value between 0 and 1. When $h = 1.0$, the Pareto-optimal front arises [16]. The Pareto-optimal front is where the non-dominated solutions from the MOGA technique reside, and the solutions are well dispersed in the solution space (Figure 4). This demonstrates how the MOGA technique for this test problem converges to the Pareto-optimal front.

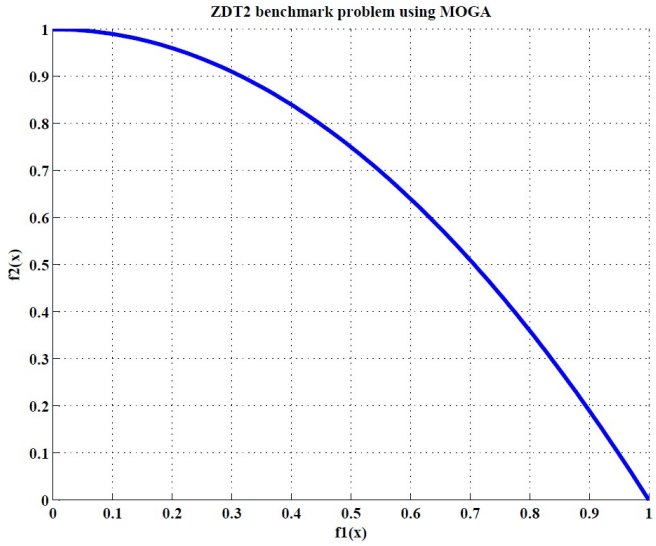

Figure 4: ZDT2 benchmark problem using MOGA

7 Results and discussion

As a test problem, the real-world project network shown in Figure 5 [3] is appropriately modified to include a non-linear relationship between activity time and cost. The problem is then tackled using the proposed approach (LI-MOGA). Table 1 also displays the various aspects of each activity.

Table 1: Alternatives of test problem

ID	Time	Cost	ID	Time	Cost
1	14	2400	10	15	450
1	15	2150	10	22	400
1	16	1900	10	23	390
1	18	1750	10	27	345
1	21	1500	10	28	330
1	23	1300	10	30	325
1	24	1200	10	33	320
2	15	3000	11	12	450
2	17	2630	11	13	420
2	18	2400	11	14	370
2	20	1800	11	16	350
2	21	1720	11	17	305
2	23	1500	11	19	300
2	25	1000	11	20	300
3	15	4500	12	22	2000
3	17	4415	12	24	1750
3	19	4220	12	25	1690
3	22	4000	12	27	1525
3	25	3730	12	28	1500
3	30	3375	12	29	1200
3	33	3200	12	30	1000
4	12	45000	13	14	4000
4	13	44300	13	15	3795
4	15	38450	13	16	3500
4	16	35000	13	18	3200
4	18	33700	13	21	2750
4	19	32400	13	23	2155
4	20	30000	13	24	1800
5	22	20000	14	9	3000
5	24	17500	14	10	2930
5	25	16400	14	12	2825
5	26	15900	14	14	2605
5	27	15700	14	15	2400
5	28	15000	14	17	2295
5	30	10000	14	18	2200
6	14	40000	15	10	6525
6	16	39200	15	13	5990
6	17	34500	15	14	4500
6	18	32000	15	16	3500
6	20	27700	15	17	3355
6	22	20300	15	18	2600
6	24	18000	15	20	1930
7	9	30000	16	20	3000
7	11	27200	16	22	2000
7	13	26100	16	24	1750
7	14	25600	16	26	1685
7	15	24000	16	28	1500
7	17	22300	16	29	1385
7	18	22000	16	30	1000
8	14	220	17	14	4000
8	15	215	17	16	3700
8	16	200	17	17	3455
8	17	190	17	18	3200
8	21	167	17	21	2780
8	23	150	17	23	2335
8	24	120	17	24	1800
9	15	300	18	9	3000
9	16	280	18	10	2900
9	18	245	18	12	2790
9	21	175	18	14	2565
9	22	135	18	15	2400
9	24	115	18	16	2315
9	25	100	18	18	2200

Initial population (n_p), mutation rate (p_m), and crossover rate (c_r) are defined, respectively, as 540, 0.02,

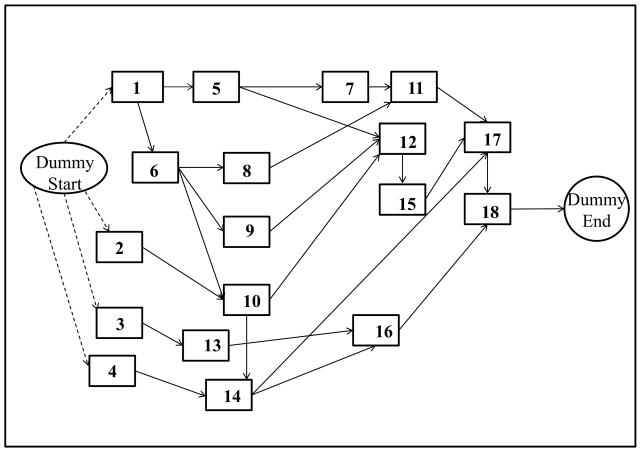

Figure 5: Network of test problem

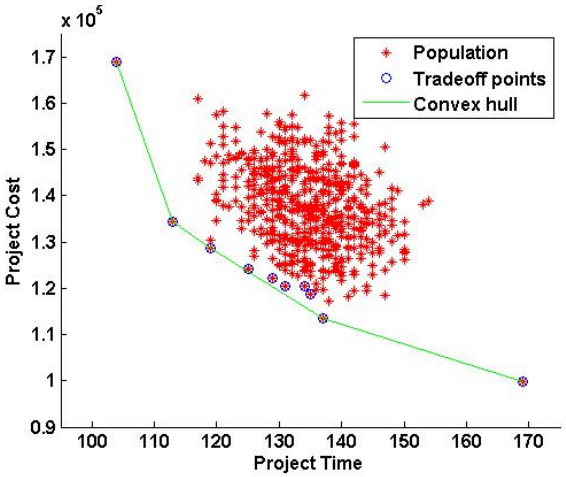

Figure 6: Initial generation

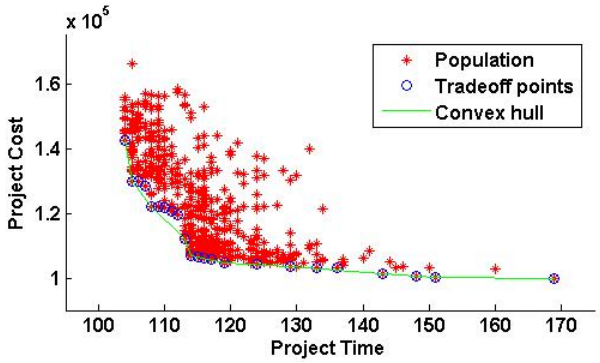

Figure 7: Intermediate generation

and 1.0. Additionally, as 5 consecutive iterations were determined to be a sufficient amount, the search is configured to end when the trade-off profile does not change during that time. Initial population n_p of 540 strings is randomly selected. An exhaustive experiment is used to select an initial population, and 540 is ultimately determined to be a suitable number. Because the population's diversity is preserved and the convergence period is short, the population is scattered across all feasible spaces. Figure 6 illustrates how n_p is dispersed throughout the solution space and does not concentrate in any particular area. The new population shifts in the direction of axes with intermediate improvements in trade-off profile and convex hull are shown in Figure 7. A clear improvement in trade-off points and convex hull across generations can be seen in the best-attained trade-off points and their convex hull (Figure 8). This profile is determined to be the best TCT profile according to LI-MOGA search criteria because the trade-off profile does not improve further.

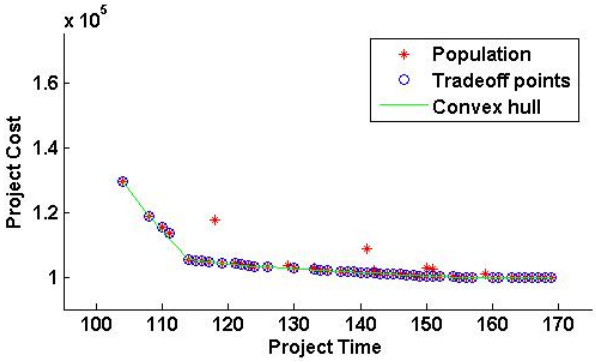

Figure 8: Final generation

8 Conclusion

LI-MOGA can identify optimal activity allocation using any time-cost functions, and it is not constrained by the form of the activity time-cost relationship. In order to validate the performance of the MOGA utilized, two common test problems are effectively addressed. LI's versatility and higher modeling capacity are demonstrated by providing various cost-evaluation options for each activity. The integrated approach uses a novel combination of LI and MOGA to address non-linear multi-objective optimization problems. This is a powerful and efficient multi-objective optimization technique that can search for a near-optimal/optimal realistic TCT profile after exploring a small search space. The proposed approach can assist in real-time to choose the best alternative among non-linear TCT profiles to execute the real world projects.

References

[1] John H. Holland. Adaptation in natural and artificial systems. Ann Arbor: The University

of Michigan Press, Michigan, pp. 1–23, 1975. https://doi.org/10.1145/1216504.1216510

[2] David E. Goldberg. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, p. 673, 1989. https://doi.org/10.5860/choice.27-0936

[3] Chung-Wei Feng, Liang Liu, and Scott A. Burns. Using genetic algorithms to solve construction time-cost trade-off problems. Journal of Computing in Civil Engineering, 11(3):184–189, 1997. https://doi.org/10.1061/(asce)0887-3801(1997)11:3(184)

[4] Duzgun Agdas, David J. Warne, Jorge Osio-Norgaard, Forrest J. Masters. Utility of genetic algorithms for solving large-scale construction time-cost trade-off problems. Journal of Computing in Civil Engineering, 32(1), 2018. https://doi.org/10.1061/(asce)cp.1943-5487.0000718

[5] S.M. Reza Alavipour and David Arditi. Time-cost tradeoff analysis with minimized project financing cost. Automation in Construction, 98:110–121, 2019. https://doi.org/10.1016/j.autcon.2018.09.009

[6] Bhupendra Kumar Pathak, Sanjay Srivastava, and Kamal Srivastava. Neural network embedded multiobjective genetic algorithm to solve non-linear time-cost tradeoff problems of project scheduling. Journal of Scientific and Industrial Research, 67, 2008.

[7] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80:8091–8126, 2021. https://doi.org/10.1007/s11042-020-10139-6

[8] Gülçağ Albayrak. Novel hybrid method in time–cost trade-off for resource-constrained construction projects. Iranian Journal of Science and Technology, Transactions of Civil Engineering, 44(4):1295–1307, 2020. https://doi.org/10.1007/s40996-020-00437-2

[9] A.P. Chassiakos and G. Rempis. Evolutionary algorithm performance evaluation in project time-cost optimization. Journal of Soft Computing in Civil Engineering, 3(2):16–29, 2019.

[10] S.S. Venkatesh and Deepak Mishra. Variable search space converging genetic algorithm for solving system of non-linear equations. Journal of Intelligent Systems, 30(1):142–164, 2020. https://doi.org/10.1515/jisys-2019-0233

[11] Osama Farouk Hassan, Amani Jamal, Sayed Abdel-Khalek. Genetic algorithm and numerical methods for solving linear and nonlinear system of equations: a comparative study. Journal of Intelligent & Fuzzy Systems, 38(3):2867–2872, 2020. https://doi.org/10.3233/jifs-179572

[12] L. Haldurai, T. Madhubala, and R. Rajalakshmi. A study on genetic algorithm and its applications. International Journal of Computer Sciences and Engineering, 4(10):139, 2016.

[13] Deepak Sharma and Sanjay Tiwari. Genetic algorithm-based optimization model for time-cost trade-off for construction project. International Journal for Research in Applied Science & Engineering Technology, 2023. https://doi.org/10.22214/ijraset.2023.48621

[14] Bhupendra Kumar Pathak and Sanjay Srivastava. Effects of project uncertainties on nonlinear time-cost tradeoff profile. Iranian Journal of Fuzzy Systems, 12(4):79–100, 2015.

[15] E.T. Whittaker and G. Robinson. The calculus of observations: a treatise on numerical mathematics. Journal of the Royal Statistical Society, 87(2):291–293, 1924. https://doi.org/10.2307/2341229

[16] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation, 8(2):173–195, 2000. https://doi.org/10.1162/106365600568202

[17] H. Jeffreys and B.S. Jeffreys. Methods of mathematical physics. Cambridge University Press, 1999. https://doi.org/10.1017/cbo9781139168489

[18] R. Bridges. Programming for mathematicians. The Mathematical Gazette, 2000. https://doi.org/10.2307/3620810