

Design and Implementation of a Genetic Framework for Project Scheduling Problem

Anum Kaleem¹✉, Saima Jawad¹, Urooj Rafiq²

¹Department of Computer Science, Bahria University, Islamabad, Pakistan

²Department of Information Technology, S&P Global Market Intelligence, Islamabad, Pakistan

anumkleem.buic@bahria.edu.pk; saima@bahria.edu.pk; uroojrafiq@hotmail.com

Corresponding Author: *anumkleem.buic@bahria.edu.pk*

Abstract

The quality of every project is determined by time, cost and scope – the triple constraint. Change in one factor incontrovertibly affects the others and attempting to minimize the cost and duration may prove to be an arduous task. Therefore, a successful project necessitates effective utilization of resources. To facilitate the project planning process, an automated intelligent tool is required to generate an optimal project schedule. Research has shown that genetic algorithms (GAs) can be used as an optimization tool in the project management domain. The main objective of this research is to investigate the use of GAs to enhance the efficacy of the project planning process by aiming to resolve time and cost conflicts. The research also includes a study of the effect of genetic environment parameters (population size, crossover rate, mutation probability, number of generations) on the solution efficiency. The paper describes the design and object oriented implementation of a genetic framework for the solution of the project scheduling problem. The proposition has been implemented in the C# language using Microsoft's dot Net platform. The test cases depict that optimal project cost and task duration are achieved as generation size increases. In conclusion, an increase in the permutations of project schedule possibilities increases the chances to achieve minimum project cost.

Article History

Manuscript Received
March 27, 2025

First Revision
May 8, 2025

Final Acceptance
June 2, 2025

Keywords: Genetic Algorithm; Project Scheduling Problem; Project Planning; Project Management

1 Introduction

One of the fundamental project management challenges is effectively scheduling tasks and allocating resources. Attempting to minimize both the cost and duration may prove to be an arduous task as reducing cost increases the project duration and vice versa.

Planning and scheduling requires integration of data pertaining to task details, resource constraints and the relationships among them. Depending on the kind of project, a variety of tasks may be included. For example, project planning involves requirements gathering, task analysis, design, development and testing and integration of modules. The resources employed may be both renewable and non-renewable encompassing manpower, machines, budget and raw materials. Effective assignment and management of the tasks and resources can make the difference between products that hit or miss the market.

Thus, the optimization and automation of task scheduling and resource allocation is essential for good results. To solve this problem a multitude of exhaustive search methods have been used which may require extensive execution time as size of the problem increases. Genetic Algorithms (GAs) can provide an efficient and near optimal schedule for projects.

This paper is divided into seven main sections. Section 2 presents the existing work in genetic algorithms along with the problems in project and resource scheduling. Section 3 discusses the proposed genetic framework and section 4 elaborates on the application scenario. Section 5 contains the discussion of the acquired results. Future considerations are discussed in section 6 and section 7 concludes the research.

2 Background Study

The core objectives of project scheduling are to minimize the cost and duration of the project and to minimize the number of products that are delivered late [1]. For effective project scheduling, it is necessary to assign the tasks with minimum number of resources to ensure completion in minimum duration. This complex multi-objective management problem belongs to the category of NP-hard problems [2, 3]. In order to find optimal solutions for these problems researchers have formulated numerous approaches including Genetic Algorithms (GAs), a branch of Evolutionary Computing. GAs are population-based search algorithms that reduce the possibility of finding the local optima because of the multiple search points. GAs can also be used to evolve polynomial time solutions for NP-hard problems [4]. A myriad of studies have been conducted on the use of GAs for solving the Project Scheduling Problem (PSP). A hybridized GA which integrates the Priority Rule-based Parallel Schedule Generation Scheme (PRPSGS) with conventional GA has been proposed for the Resource Constrained Project Scheduling Problem (RCPSP) with multi-skilled workforce [5]. Amr Kandil and Khaled El-Rayes implemented a parallel multi-objective genetic algorithm capable of optimizing large scale construction projects [6]. A hybrid greedy GA to reduce the make span of a project and production of good quality schedules has been proposed by Delgoshaei et al [7]. For optimization of RCPSP from different domains Rajeevan et al used GAs [8]. Findings have shown GA to be capable of searching the best known solutions from the problem space. In a study conducted by Sebt et al a Multi-Mode Resource Constraint Problem (MMRCPSP) was used to find the feasible mode and starting time for project scheduling. Here the authors have enhanced GA with an efficient fitness function and mode list based mutation operator [9]. The results show that using GA reduces the time duration for these projects. Sebt et al have proposed three different variants for GAs namely: Standard GA, Stud GA and Jumping Gene. In particular, the results show the effectiveness of Stud GA in solving RCPSP [10]. Xin Shu et al discuss a model taking into consideration priority of activities and a storage adjacency matrix to solve resource constraint multi project scheduling problem (RCMPSP). They find a solution for medical resource scheduling to enhance the feasibility of the problem solution [11]. Similarly, Firoz Mahmud et al propose a GA to solve heterogonous project scheduling problem which indicates the need of optimization of PSP [12]. In [13] Huu Dang Quoc et al perform a comparative analysis between GA, Greedy Search and differential evolution algorithm. The result suggest that differential evolution algorithm is an enhanced model for solving PSP [13].

In short, many advanced GA techniques have been used to solve PSP, RCPSP, RCMPSP, PRPSGS and MMRCPSP successfully. However, there is a need to implement an automated tool based on these sophisticated techniques. The main objective of this research is to investigate application of GAs for the solution of PSP. The proposed genetic framework is designed using object oriented paradigm and C# language and the dot NET platform are used for the implementation.

3 Genetic Project Scheduling Framework

A GA evolves the solution of a problem. The algorithm begins with an initial set of solutions called population (represented by chromosomes – a collection of genes). Fitness of the chromosomes is calculated on the generation of the population. Subsequently, offspring are formed from parent solutions

which have been selected according to their fitness. Every solution has an associated quality value which represents its chances to succeed. Crossover and Mutation are two genetic operators which are applied next. Crossover is implemented to form a new child by exchanging the information of two parent chromosomes. It is expected that the new population will be better than the old one. Mutation refers to arbitrary change in the information of a randomly selected gene from the population. Until certain conditions (such as the number of populations or improvement of the best solution) are satisfied, the entire process is repeated

Different environment parameters influence the efficiency of a GA process. Among these, the size of chromosome population size is the most important factor. A larger population size may increase the probability of computing the global optimum but on the other hand, the processing time increases for larger population. Crossover Probability refers to the number of times a crossover will be performed. The offspring is an exact copy of the parents in the absence of a crossover. In the presence of a crossover, however, the offspring is a mixture of parts of the parents' chromosome. That is to say, if the probability of the crossover is 100%, then all offspring are different from their parents. If the crossover probability is 0%, whole new generation is an exact copy of chromosomes from the parents. Mutation Probability refers to how frequently parts of chromosome are mutated. If there is no mutation, the offspring, taken after crossover (or copy), is unchanged. If mutation does in fact occur, part of chromosome is changed. Thus, in case of a 100% mutation probability, the entire chromosome is changed, whereas, if the probability is 0%, the chromosome remains unchanged. Mutation prevents falling of GA into local extreme, however, frequent occurrence should be avoided because that will in fact change GA to random search.

3.1 Solution Methodology

Figure 1 illustrates the devised methodology for the implementation of the genetic framework. First of all, the user inputs the genetic environment parameters and the project plan requirements. In order to find the fittest schedule, GA functions like crossover, mutation and fitness are applied on the initial requirements. Until a project schedule is formulated and a terminating condition is met, this process continues.

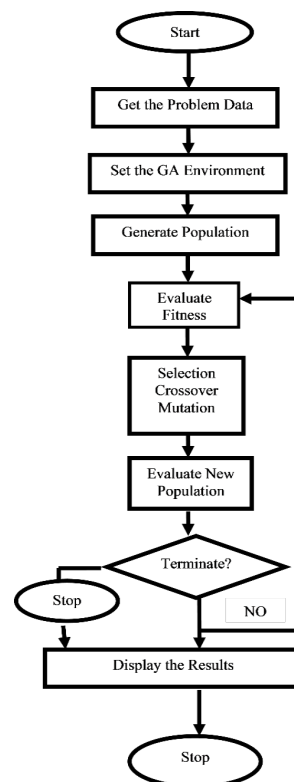


Figure 1: Solution methodology

Different states of the genetic algorithm are explained in Figure 2 where first population is generated according to the population size. Fitness of each chromosome is calculated. Chromosomes are arranged in ascending order of their fitness value. Two-point crossover is performed and mutation probability is added to all the less fit chromosomes. Chromosomes with acceptable fitness values are placed in the new generation. Processing is terminated till best fit chromosome or schedule with optimal/minimum project cost is obtained.

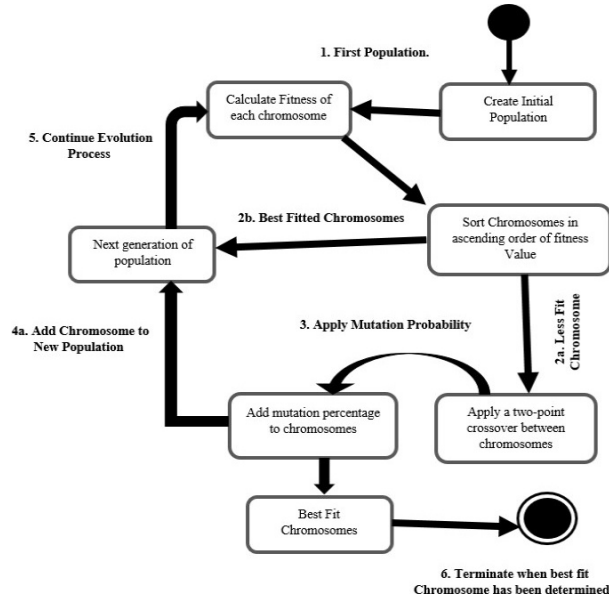


Figure 2: State diagram of solution methodology

3.2 Object Oriented Structure

Figure 3 illustrates the object oriented structure of the genetic framework. The structure comprises of gene, chromosome, population and generations. The smallest component of the structure is a gene. A gene contains the fundamental information about the problem. A chromosome is a schedule formed considering employee and project details. A specified population size determines the generation of multiple chromosomes. Various population generations are generated according to the size of the generation. Finally, until an optimal schedule is attained, fitness, crossover and mutation functions are applied on the project schedule. Different classes have been constructed in order to implement object oriented GA structure.

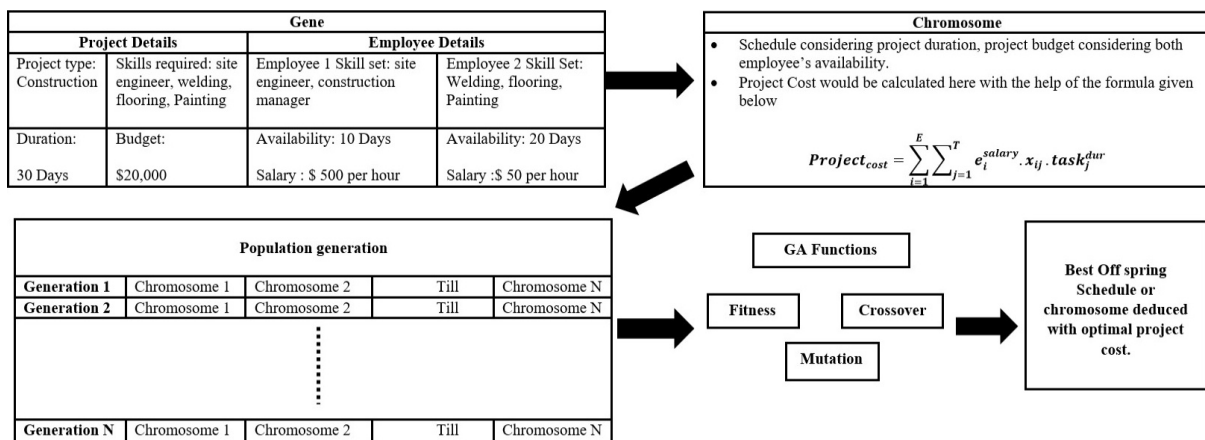


Figure 3: Object oriented GA structure

3.3 Genetic Algorithm

1. Set the GA environment parameters: population size p , generation size g , crossover rate c and termination criteria s .
2. Initialize *employee class* consisting of attributes such as *salary*, *experience in years*, *availability* and *employee skill set* [].
3. Initialize *project details class* consisting of attributes such as *title of project*, *project skill set* [], *number of employees required* and *minimum duration for task completion*.
4. Initialize *gene class* by initializing required *employee class* and *project detail class objects*.
5. Initialize an array of *chromosome class* $C[]$ with respect to the population size p . In each chromosome perform the following:
 - a. Compare *employee skillset* from *employee class* and *project skill set* from *project detail class* to formulate a *schedule*.
 - b. Calculate *project cost of each schedule*.
 - c. Calculate *fitness value* of each schedule
6. Arrange population of chromosomes in ascending order of their fitness value.
7. Initialize an array of *generation class* $G[]$ with respect to *generation size* g .
8. Calculate *fitness threshold value*.
9. Perform *crossover* using *cross-over rate* c on chromosomes with project cost greater than *fitness threshold*.
10. Perform *mutation* on chromosomes with project cost greater than fitness threshold.
11. Continue till *best fit schedule* with *minimum project cost* is obtained.

3.3.1 Population Generation

The initial population of *schedule (chromosome)* and *generations of population* $G[]$ are generated according to the generation size g . One generation consists of *multiple variants of schedules* with different project costs. A schedule is formed according to *employee availability*, *salary*, *skillset* and *project skill requirement*. Each schedules *project cost varies* in accordance to the *assignment of project tasks* and *employee skill set*.

3.3.2 Fitness Evaluation

The fitness of a schedule is computed using minimum project cost and accumulative project time duration. In accordance to calculate the project duration, the time duration of each task being performed needs to be calculated [4].

1. Let $task_j^{dur}$ be the time period of each task. This can be computed by dividing the effort for current task performance t_j^{effort} by the accumulative degree of employee dedication $\sum_{i=1}^E x_{ij}$. This value lies between 0 and 1, if an employee is unable to carry out project tasks then x_{ij} is assigned the value of 0. The equation for project task period computation is given below:

$$task_j^{dur} = \frac{t_j^{effort}}{\sum_{i=1}^E x_{ij}} \quad (1)$$

2. Let $Project_{cost}$ be the cost of current schedule. It can be computed by multiplication of employee salary, time duration of each task and employee dedication [4]. The mathematical equation of which is stated below:

$$Project_{cost} = \sum_{i=1}^E \sum_{j=1}^T e_i^{salary} \cdot x_{ij} \cdot task_j^{dur}, \quad \sum_{i=1}^E x_{ij} > 0 \quad (2)$$

3.3.3 Cross-over Function

A two dimensional single point crossover is applied where, rows and columns of parents are swapped on the basis of random selection [4]. Figure 4 below explains the above mentioned process in detail. In a way similar to that of a coin toss, the crossover is performed in order to determine which parent gene would be passed to the offspring. Until the crossover probability c is reached, alternate schedules from the present generations are selected to be placed in the new offspring.

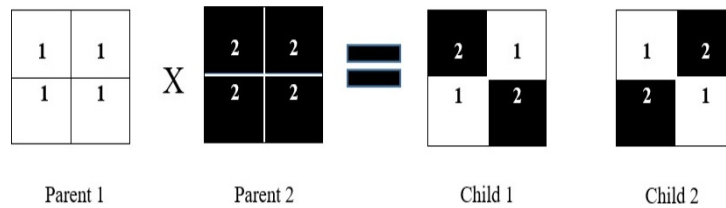


Figure 4: 2D Single point crossover

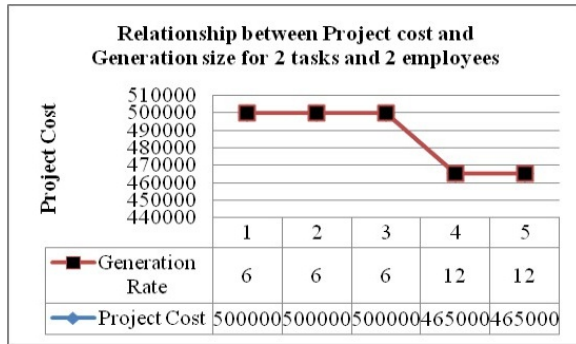
4 GA Framework Application Design

To implement the GA framework, an application titled “GAP Scheduler” was designed and developed utilizing C# language dot NET platform. The application requests the user to enter task and resource skill set information. This data is entered in separate interfaces and is subsequently stored in a local repository database for optimal schedule determination. User then specifies GA parameters which are cross-over rate, mutation rate, population size and generation size. If user chooses not to specify GA parameters, then default parameter values are applied. Finally, the optimal schedule is displayed by the application in the form of a Gantt chart.

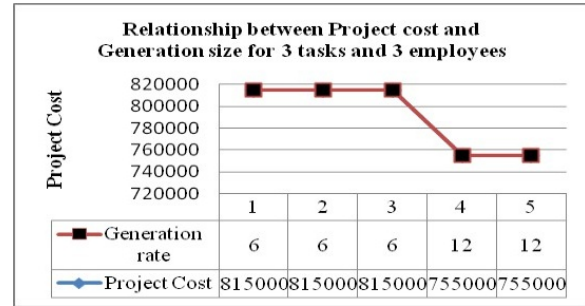
5 Results and Analysis

The relationship between framework input variables, GA environment parameters and the output variables is displayed in Table 1. The input parameters include the total employees E_{Total} , total project tasks P_{Tasks} , the skills each task requires PT_{skills} , and furthermore, the skills each employee possesses Emp_{skills} . The generation size G_{size} , population size P_{size} , crossover rate C_{rate} and mutation rate M_{rate} are the GA environment parameters. The last two columns represent the project cost $Project_{cost}$ and the time duration $Task_{dur}$ which are the two output variables calculated for each project schedule.

For the analysis, test cases with different number of employees, tasks and GA parameters are generated. The results depict that the more the permutations of a schedule the more the chances of achieving a schedule with minimum cost and time. Figure 5a and 5b illustrate this relationship proving that project cost decreases by increasing the generation size. The results also show that the population size and crossover rate have no effect on project cost.



(a) Relationship between project cost and generation size for two task and two employees



(b) Relationship between project cost and generation size for three tasks and three employees

Figure 5: Relationship between project cost and generation size

Table 1: Result of test cases

	E_{total}	P_{task}	PT_{skills}	Emp_{skills}	G_{size}	P_{size}	C_{rate}	M_{rate}	$Task_{dur}$	$Project_{cost}$
1	2	2	Task 1: 6, Task 2: 3	Emp1: 6, Emp2: 3	6	10	0.7	0.09	29	500000
2	2	2	Task 1: 6, Task 2: 3	Emp1: 6, Emp2: 3	6	20	0.7	0.09	29	500000
3	2	2	Task 1: 6, Task 2: 3	Emp1: 6, Emp2: 3	6	10	1	0.09	22	500000
4	2	2	Task 1: 6, Task 2: 3	Emp1: 6, Emp2: 3	12	20	1	0.09	22	465000
5	2	2	Task 1: 6, Task 2: 3	Emp1: 6, Emp2: 3	12	10	0.7	0.09	22	465000
6	3	3	Task1: 4, Task2: 1, Task3: 2	Emp1: 7, Emp2: 2, Emp3: 4	6	10	0.7	0.09	42	815000
7	3	3	Task1: 4, Task2: 1, Task3: 2	Emp1: 7, Emp2: 2, Emp3: 4	6	20	0.7	0.09	49	815000
8	3	3	Task1: 4, Task2: 1, Task3: 2	Emp1: 7, Emp2: 2, Emp3: 4	6	10	1	0.09	49	815000
9	3	3	Task1: 4, Task2: 1, Task3: 2	Emp1: 7, Emp2: 2, Emp3: 4	12	10	0.7	0.09	42	755000
10	3	3	Task1: 4, Task2: 1, Task3: 2	Emp1: 7, Emp2: 2, Emp3: 4	12	20	1	0.09	42	755000

6 Future Considerations

To manage conflicting requirements in the project scheduling problem the proposed framework can further be improved. The current approach is focused on project scheduling for a single project. This framework would be more effective if it could manage finding optimal schedule cost for multiple projects simultaneously. It would provide project managers efficient solutions for PSP with minimum permutations collectively. Another consideration would be the use of priority based crossover instead of single point crossover [14]. By incorporating priority based crossover project managers would be able to set precedence to which gene is being passed to the offspring instead of random selection. Priority based crossover example is given in the Table 2. Here user can select which set of activity should be passed over to the new offspring. In such crossover user would be able to input crossover task priority instead of crossover rate.

Table 2: Priority based crossover

Parent 1								
Priority	1			2			3	
Task/Activity List	1	3	6	5	7	4	8	2
Parent 2								
Priority	1			2			3	
Task/Activity List	1	2	4	6	7	5	3	8
<i>Exchange priority no 2 from both parents lists</i>								
Child 1								
Priority	1			2			3	
Task/Activity List	1	3	4	6	7	5	8	2
Child 2								
Priority	1			2			3	
Task/Activity List	1	2	6	5	7	4	3	8

7 Conclusion

In practical life, projects mostly have strict constraints of time and cost. Automation of finding optimal solution to PSP can save project resources. In this research work we implemented a GA based Project Scheduling application called GAP Scheduler to facilitate the project managers. The application will aid them to build an efficient schedule consisting of minimal cost and duration. Moreover, the utilization of resources is automatically optimized. This GA based tool can be used in the project management domain for the optimal assignment of workforce to the tasks. This scheduler is a step forward in the application of evolutionary algorithms for the solution of real world problems. The efficiency of the existing implementation can be further increased by using multithreading for the genetic computations. In the future we plan to enhance the functionality of application by incorporating provision of multiple projects' scheduling simultaneously.

References

- [1] M. B. Wall, "A genetic algorithm for resource-constrained scheduling," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 1996.
- [2] D. Sundar, B. Umadevi, and K. Alagarsamy, "Multi objective genetic algorithm for the optimized resource usage and the prioritization of the constraints in the software project planning," *International Journal of Computer Applications*, vol. 3, no. 3, pp. 1–4, 2010.
- [3] B. K. T. Trzaskalik, "Genetic algorithm in multi-criteria multi-project scheduling problem," pp. 2–3.
- [4] E. Alba and J. F. Chicano, "Software project management with GAs," *Information Sciences*, vol. 177, no. 11, pp. 2380–2401, 2007.
- [5] C. Liu and S. Yang, "A hybrid genetic algorithm for integrated project task and multi-skilled workforce scheduling," 2011, vol. 6, pp. 2187–2194.
- [6] A. Kandil and K. El-Rayes, "Parallel genetic algorithms for optimizing resource utilization in large-scale construction projects," *Journal of Construction Engineering and Management*, vol. 132, no. 5, pp. 491–498, 2006.

- [7] A. Delgoshaei, M. K. M. Ariffin, B. T. H. T. B. Baharudin, and Z. Leman, "Minimizing makespan of a resource-constrained scheduling problem: A hybrid greedy and genetic algorithms," *International Journal of Industrial Engineering Computations*, vol. 6, no. 4, pp. 503–520, 2015.
- [8] M. Rajeevan and R. Nagavinothini, "Time optimization for resource-constrained project scheduling using meta-heuristic approach," *International Journal of Science, Engineering and Technology Research*, vol. 4, no. 3, pp. 606–609, 2015.
- [9] M. H. Sebt, M. R. Afshar, and Y. Alipouri, "An efficient genetic algorithm for solving the multi-mode resource-constrained project scheduling problem based on random key representation," *International Journal of Supply and Operations Management*, vol. 2, no. 3, pp. 905–924, 2015.
- [10] M. H. Sebt, M. H. F. Zarandi, and Y. Alipouri, "Genetic algorithms to solve resource-constrained project scheduling problems with variable activity durations," *International Journal of Civil Engineering*, vol. 11, no. 3, 2013.
- [11] X. Shu, Q. Su, Q. Wang, and Q. Wang, "Optimization of resource-constrained multi-project scheduling problem based on the genetic algorithm," in *2018 15th International Conference on Service Systems and Service Management (ICSSSM)*, 2018, pp. 1–6.
- [12] F. Mahmud, F. Zaman, R. Sarker, and D. Essam, "Heuristic embedded genetic algorithm for heterogeneous project scheduling problems," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020.
- [13] H. D. Quoc, L. N. The, C. N. Doan, and T. P. Thanh, "New effective differential evolution algorithm for the project scheduling problem," in *2020 2nd International Conference on Computer Communication and the Internet (ICCCI)*, 2020, pp. 150–157.
- [14] S. U. Kadam and S. U. Mane, "A genetic-local search algorithm approach for resource constrained project scheduling problem," in *2015 International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2015, pp. 841–846.