UIMP
Universidad Internacional Menéndez Pelayo

AEPIA
ASOCIACIÓN ESPAÑOLA PARA LA INTELIGENCIA ARTIFICIAL

Statistical Analysis of Genetic Algorithm Performance on the Multidimensional Knapsack Problem

Máster Universitario en Investigación en Inteligencia Artificial

Resolución de Problemas con Metaheurísticos

Javier Vela Tambo

January 10, 2025

Video Presentation:
`https://www.youtube.com/watch?v=7Z3a22ygyIs`

GitHub Repository:
`https://github.com/javiervela/KnapsackGeneticAlgorithm`

1 Introduction

Genetic Algorithms (GAs) are optimization techniques inspired by natural selection and genetics, where populations of solutions evolve over generations. GAs use the principles of selection, crossover, mutation, and replacement (*genetic operators*). GAs have been applied to a wide range of problems in various domains. They are mainly used in combinatorial optimization problems, where the search space is discrete and the goal is to find the best combination of elements.

The objective of this study is to evaluate the performance of a GA applied to the Multidimensional Knapsack Problem, with relevance in fields such as logistics, resource allocation, and financial decision-making. The focus of the study is on analyzing the impact of crossover and mutation probabilities on the GA's performance. Correctly choosing these parameters is crucial in the GA's execution, affecting the quality of the solutions found. The study considers two stopping criteria: a fixed number of evaluations and finding the optimal solution.

This report introduces the Multidimensional Knapsack Problem (MKP) (Section 2) and the problem instances used. The methodology, including genetic operators, parameter selection, and stopping criteria, is presented in Section 3. Section 4 describes the experiments and data collection. The results are analyzed in Section 5, focusing on fitness values and execution times. Finally, Section 6 summarizes the findings and discusses limitations. The appendix includes additional data and figures for the performance analysis.

2 Multidimensional Knapsack Problem

The *Knapsack Problem* is a well-known combinatorial optimization problem there are a set of items, each with a weight (a resource) and value. The goal is to select a subset of items, maximizing the total value without exceeding the weight capacity of the knapsack. The Knapsack Problem is considered the "easiest" NP-hard problem [6, 7].

One of the variants of the Knapsack Problem is the *Multidimensional 0-1 Knapsack Problem*[4], which extends the Knapsack Problem to multiple dimensions. In the MKP, each item consumes multiple resources (dimensions), each with their capacity constraint. The MKP is a more complex problem than the classical Knapsack Problem, as it involves multiple constraints.

2.1 Problem Definition

Given:

- A set of n items, each with a profit p_j and resource requirements r_{ij} for m constraints,
- m knapsack constraints, each with a maximum capacity b_i,
- A binary decision variable x_j, where $x_j = 1$ if item j is included in the knapsack, and $x_j = 0$ otherwise.

The goal is to maximize the total profit while satisfying all resource constraints. The MKP can be formulated as follows [5]:

$$\text{maximize} \quad Z = \sum_{j=1}^{n} p_j x_j \tag{1}$$

subject to the constraints:

$$\sum_{j=1}^{n} r_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \tag{2}$$

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n \tag{3}$$

2.2 Problem Instances

The problem instances used in the study are the seven MKP instances[1] provided by the OR-Library [1]. These instances vary in the number of items, dimensions, and constraints. The seven MKP instances are provided in a single file[2].

The different problem instances are characterized by the number of items n and the number of constraints m, which determine the complexity of the problem. The instances used in the study are summarized in Table 1.

Instance	Items (n)	Constraints (m)	Optimal Value
0	6	10	3800
1	10	10	8706.1
2	15	10	4015
3	20	10	6120
4	28	10	12400
5	39	5	10618
6	50	5	16537

Table 1: Summary of the MKP problem instances used in the study.

[1] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html`
[2] `https://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/mknap1.txt`

3 Methodology

3.1 Genetic Operators

The study's GA employs a specific method for each of the standard genetic operators: selection, crossover, mutation, and replacement. The chosen methods for this study are described below:

- **Selection**: The *Binary Tournament Selection* operator is used to select two parents for reproduction. In this method, two individuals are randomly selected from the population, and the one with the better fitness is chosen.
- **Crossover**: The *Single-point Crossover* operator is used to generate a single child per generation. A random crossover point is selected, and the segments after this point are swapped between two parents to create offspring. This operator is applied with a specified probability.
- **Mutation**: The *Bit-flip Mutation* operator introduces diversity by flipping bits in the binary string representation of the child. Each bit has a specified probability of being flipped.
- **Replacement**: An *Elitist Replacement* strategy is used. In this strategy, the newly generated child is included in the population, and then the worst individual is removed, ensuring that the best solutions are preserved.

3.2 Parameter Selection

The performance of the GA is influenced by the crossover and mutation probabilities. These parameters affect the balance between exploration and exploitation. Finding the right balance between these probabilities is crucial for the GA's performance.

3.2.1 Crossover Probabilities

The crossover probability determines the likelihood of applying the crossover operator to generate offspring. The crossover probability values explored in this study are defined as follows:

$$p_c \in \{0.1, 0.3, 0.5, 0.7, 0.9, 1.0\} \tag{4}$$

A higher crossover probability promotes exploration by combining different parts of parent solutions, leading to new solutions. However, if too high, it may degrade good solutions.

3.2.2 Mutation Probabilities

The mutation probability determines the likelihood of applying the mutation operator to each bit of the binary string representation of the offspring. The mutation probability values explored in this study are defined as follows:

$$p_m \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.5\} \tag{5}$$

Higher mutation probabilities introduce more diversity in the solutions, helping to explore the search space. However, if too high, it can prevent convergence of the GA.

3.3 Stopping Criteria

A stopping criterion is the condition that determines when the GA algorithm should terminate. It is important because it ensures that the algorithm does not run indefinitely and provides a balance between computational effort and solution quality. Two stopping criteria are considered in this study to evaluate the performance of the GA.

3.3.1 Maximum Number of Evaluations

The first stopping criterion is based on a fixed number of evaluations. The GA will stop after a specified number of evaluations are performed, regardless of whether the optimal solution has been found. In the study, the number of evaluations is set to $N_e = 1000$.

3.3.2 Finding the Optimum Solution

The second stopping criterion is based on finding the optimal solution ($N_e = *$). The GA will stop when the best fitness value is equal to the known optimal value for the MKP instance.

4 Experiments

4.1 Implementation Details

The GA used in the study is based on the Java implementation by Francisco Chicano. The *SimpleEvolutionaryAlgorithm*[3] implementation, was forked to *KnapsackGeneticAlgorithm*[4] and adapted to solve the MKP.

The *MultidimensionalKnapsackProblem* class was developed to adapt the GA for the MKP. This class encapsulates the MKP instance, including item profits, constraints, and capacities. The implementation leverages the existing *binary string* for representing the solutions in the population, where each bit indicates whether an item is included in the knapsack. A custom fitness function evaluates solutions based on MKP constraints, assigning a fitness value of 0 to any solution that violates a constraint.

The *MultidimensionalKnapsackProblemLoader* class was created to load the problem instances and their optimal values from the file downloaded from the OR-Library.

The base implementation includes the genetic operators described in Section 3.1. The *EvolutionaryAlgorithm* class was extended to allow setting a crossover probability (p_c). The implementation was also modified to incorporate the different stopping criteria described in Section 3.3. Specifically, a *StoppingCriterion* interface was developed, and the *OptimalSolutionCriterion* class, which implements this interface, was created to stop the GA when finding the optimal solution.

It exists the possibility that the GA does not find the optimal solution within a feasible time. Specifically, this behavior can occur when the mutation probability and crossover probability are set to low values, as the search space is not explored effectively. To address this, a timeout mechanism was implemented for the *OptimalSolutionCriterion*. The timeout value was set to 1 minute.

4.2 Data Collection for Evaluation

The *MetricsCollector* class was developed to collect data from each execution of the algorithm for the statistical analysis. The data collected includes the problem instance details, the algorithm's parameters, execution time, number of generations and evaluations, final solution, and the best solution's fitness value for each generation.

For executions that run for a large number of generations, saving the best fitness value for each generation can be expensive in terms of storage. To address this, the *MetricsCollector* saves the best fitness value only when it improves, reducing the amount of data stored.

After each execution, the data collected is stored in a JSON file for further analysis. The result files are organized by the stopping criteria, MKP instance index, crossover probability, mutation probability, and run index.

4.3 Execution Details

The experiments were conducted on a machine running Arch Linux. The system is equipped with 16 GB of RAM and an Intel Core i5-10210U CPU, which has four cores and eight threads. The Java environment used is OpenJDK 11.0.25.

For each combination of stopping criteria, crossover probability (p_c), and mutation probability (p_m), $N_r = 31$ independent runs were executed on each of the 7 (N_i) instances of the problem. The number N_r was chosen to ensure statistical significance in the results. For each stopping criteria the total number of runs is:

$$N_r \times N_i \times N_{p_c} \times N_{p_m} = 31 \times 7 \times 6 \times 6 = 7812 \tag{6}$$

[3] `https://github.com/NEO-Research-Group/SimpleEvolutionaryAlgorithm`
[4] `https://github.com/javiervela/KnapsackGeneticAlgorithm`

The total execution time for the experiments was approximately 6 minutes for the $N_e = 1000$ stopping criterion and 22.5 hours for the $N_e = *$ stopping criterion. The elevated execution time for the optimal solution criterion justifies the use of the timeout mechanism to prevent the algorithm from running indefinitely.

5 Results Analysis

In order to analyze the performance of the GA on the MKP instances, two different studies were conducted with different stopping criteria: a fixed number of evaluations ($N_e = 1000$) and finding the optimal solution ($N_e = *$).

For each problem instance and parameter combination (p_c, p_m), the mean and standard deviation of the final solution's fitness and execution time across the $N_r = 31$ runs were computed.

5.1 Finding Optimal Solution

The first study focuses on the GA executions where the stopping criterion is finding the optimal solution ($N_e = *$) or reaching the 1 minute timeout. In this study, the execution time analysis seems more relevant, as it provides insights into the algorithm's convergence behavior.

5.1.1 Solution's Fitness Analysis

The results for the mean and standard deviation of the fitness of the final solution of the GA are presented in Figure 2 and Table 2.

For all problem, the GA was generally able to find the optimal solution in all (instances 0, 2, and 3) or most runs (instances 1, 4, 5, and 6).

For the larger instances, the experiments performed with more extreme mutation probabilities ($p_m = 0.01$ and $p_m = 0.5$) had a a lower mean in the fitness values, indicating that the search space was not explored effectively. The higher standard deviation for these experiments suggests that the GA found the optimal solution in some of the runs but not in others.

On the other hand, the crossover probability did not seem to have a significant impact on the results.

5.1.2 Execution Time Analysis

The results for the mean and standard deviation of the execution time of the GA are presented in Figure 3 and Table 3.

In most problem instances (0, 2, and 3) the execution time was generally low and did not vary significantly across most parameter combinations. For problem instances 1 and 4, some parameter combinations did not find the optimal fitness value within the timeout for some runs. For the larger problem instances (5 and 6), the mean execution time was higher, finding the optimal value in a significantly longer time than for other instances and approaching the timeout in most parameter combinations.

The execution time analysis aligns with the earlier fitness value results, showing that the GA performs poorly with more extreme mutation probabilities ($p_m = 0.01$ and $p_m = 0.5$) and that the crossover probability has less impact on the results.

5.2 Fixed Number of Evaluations

The second study focuses on the GA executions where the stopping criterion is a fixed number of evaluations ($N_e = 1000$). In this study, the fitness analysis is more relevant, as it provides insights into the algorithm's parameters influence on performance in terms of the quality of the solutions found.

5.2.1 Solution's Fitness Analysis

The results for the mean and standard deviation of the fitness of the final solution of the GA are presented in Figure 4 and Table 4.

For all problem instances, the executions with more balanced mutation probabilities ($p_m \in \{0.1, 0.2, 0.3\}$) had a mean fitness value closer to the optimal value in comparison to the more extreme values ($p_m \in \{0.01, 0.05, 0.5\}$).

The crossover probability did not seem to have a such a significant impact on the results, although the experiments with higher crossover probabilities $p_c \in \{0.7, 0.9, 1.0\}$ had a slightly higher mean fitness value.

5.2.2 Execution Time Analysis

The results for the mean and standard deviation of the execution time of the GA are presented in Figure 5 and Table 5.

The analysis of the execution time for the fixed number of evaluations criterion seems to be less relevant, as the execution time is limited by the same number of evaluations for every parameter combination. However, the results show that the execution time was generally low and did not vary significantly across different parameter combinations. Any differences in execution time are likely due to the execution environment and other external factors.

5.3 Statistical Analysis for the Fixed Evaluations Criterion

In addition to the earlier exploratory analysis, a hypothesis test was performed to determine whether the mean fitness values of the GA executions, obtained with different crossover and mutation probabilities, differ significantly under the fixed number of evaluations criterion. This analysis was not performed for the criterion that finds the optimal solution, as those results were very similar regardless of the parameter combinations and did not produce meaningful differences.

The chosen non-parametric statistical test was the Wilcoxon signed-rank test[3], comparing the mean fitness values of pairs of parameter combinations across all problem instances. The null hypothesis (H_0) states that the mean fitness values of the GA executions with different parameter settings are equal, while the alternative hypothesis (H_1) states that at least one set of parameters leads to significantly different mean fitness values. When a p-value is less than the significance level ($\alpha = 0.05$), it indicates a statistically significant difference, allowing rejection of H_0.

The Benjamini-Hochberg[2] correction was applied to control the false discovery rate and avoid type I errors due to multiple comparisons. This method adjusts the p-values to ensure that the proportion of false positives among the significant results is kept below a specified threshold.

The table of p-values for all parameter combination pairs would be excessively large, so it is not included in this report. The complete results of the statistical test are available in the GitHub repository[5].

Based on the Wilcoxon signed-rank test results, only parameter combinations found to be significantly different from the others were considered further. Among these, the best combination was selected based on the highest mean fitness values. The Figure 1 shows the number of wins of each parameter combination in the Wilcoxon signed-rank test.

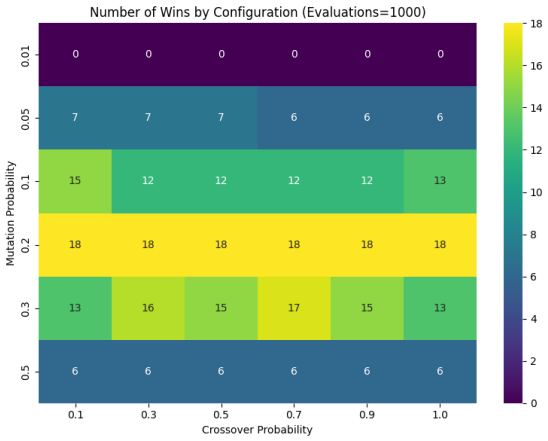

Figure 1: Heatmap of the number of wins of each parameter combination in the Wilcoxon signed-rank test.

[5] https://github.com/javiervela/KnapsackGeneticAlgorithm/blob/main/analysis/1000_evaluations/stat_test_results.csv

In the event of a tie, the parameter combination with the greatest mean fitness value was chosen. From this procedure, the combination $p_c = 0.9$ and $p_m = 0.2$ resulted in the best-performing configuration under the fixed number of evaluations criterion, consistently providing superior mean fitness values across all MKP instances tested.

6 Conclusions

The analysis of the results of the GA's execution shows that the performance of the algorithm is significantly influenced by the mutation probability. The crossover probability seems to have less impact on the results. For smaller problem instances, the GA found the optimal solution in most cases. However, for larger instances, the GA only sometimes found the optimal solution within the timeout.

The analysis for the fixed number of evaluations criterion showed that setting balanced mutation probabilities ($p_m \in \{0.1, 0.2, 0.3\}$) yields better results, providing a good balance between exploration and exploitation. Setting the crossover probability seems to have a less significant impact, although using higher values ($p_c \in \{0.7, 0.9, 1.0\}$) can be beneficial.

The statistical analysis conducted for the fixed number of evaluations criterion showed that the combination of $p_c = 0.9$ and $p_m = 0.2$ consistently provided the best results across all problem instances. This combination was found to have the highest mean fitness values and was statistically significantly different from other parameter combinations.

This study has several limitations. The scope of problem instances was limited to seven MKP instances from the OR-Library, which may not fully represent the diversity of real-world problems. Additionally, the computational resources and time constraints imposed by the 1-minute timeout may have affected the GA's ability to find optimal solutions for larger instances. The statistical analysis had many limitations and assumptions, and the results should be interpreted with caution.

References

[1] J. E. Beasley. Or-library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[2] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

[3] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[4] Arnaud Fréville. The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research*, 155(1):1–21, 2004.

[5] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Multidimensional Knapsack Problems*, pages 235–283. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[6] Bernhard Korte and Jens Vygen. *The Knapsack Problem*, pages 439–448. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[7] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., USA, 1990.

A Result Figures for Performance Analysis

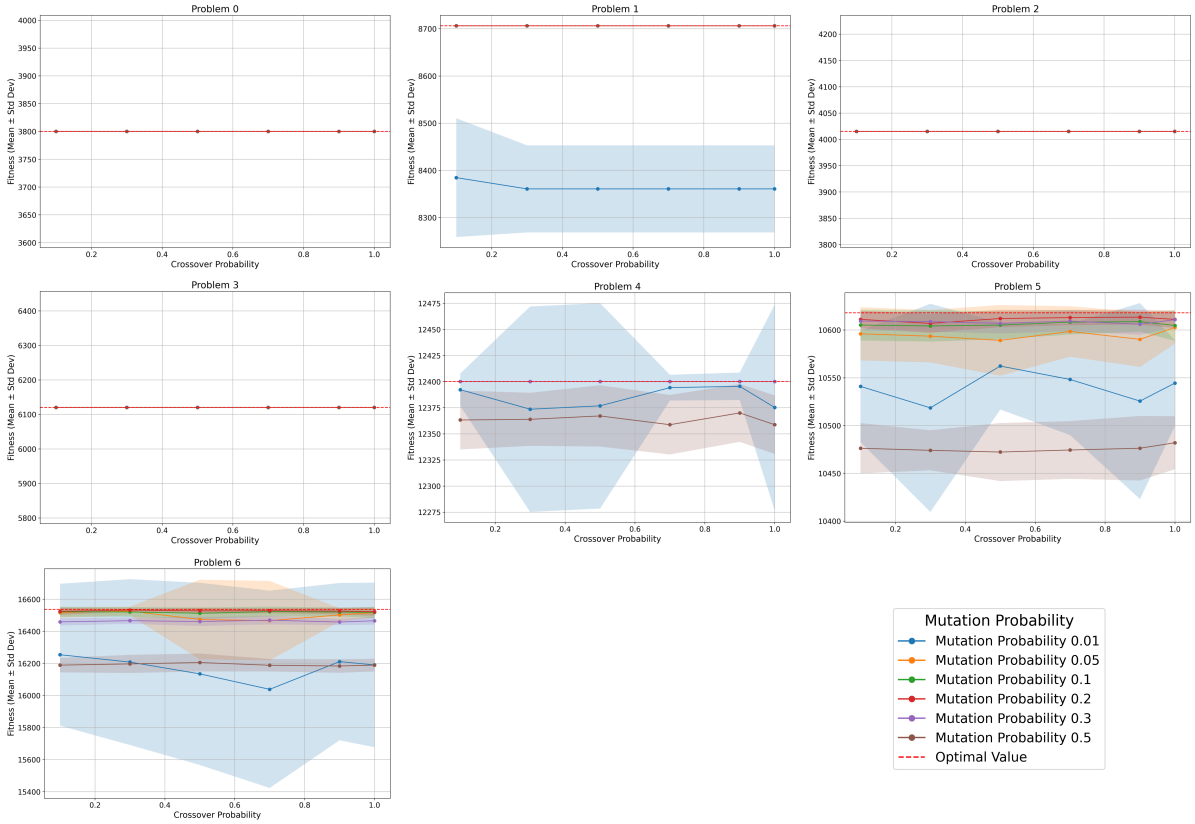

Figure 2: Mean and standard deviation of fitness with $N_e = *$.

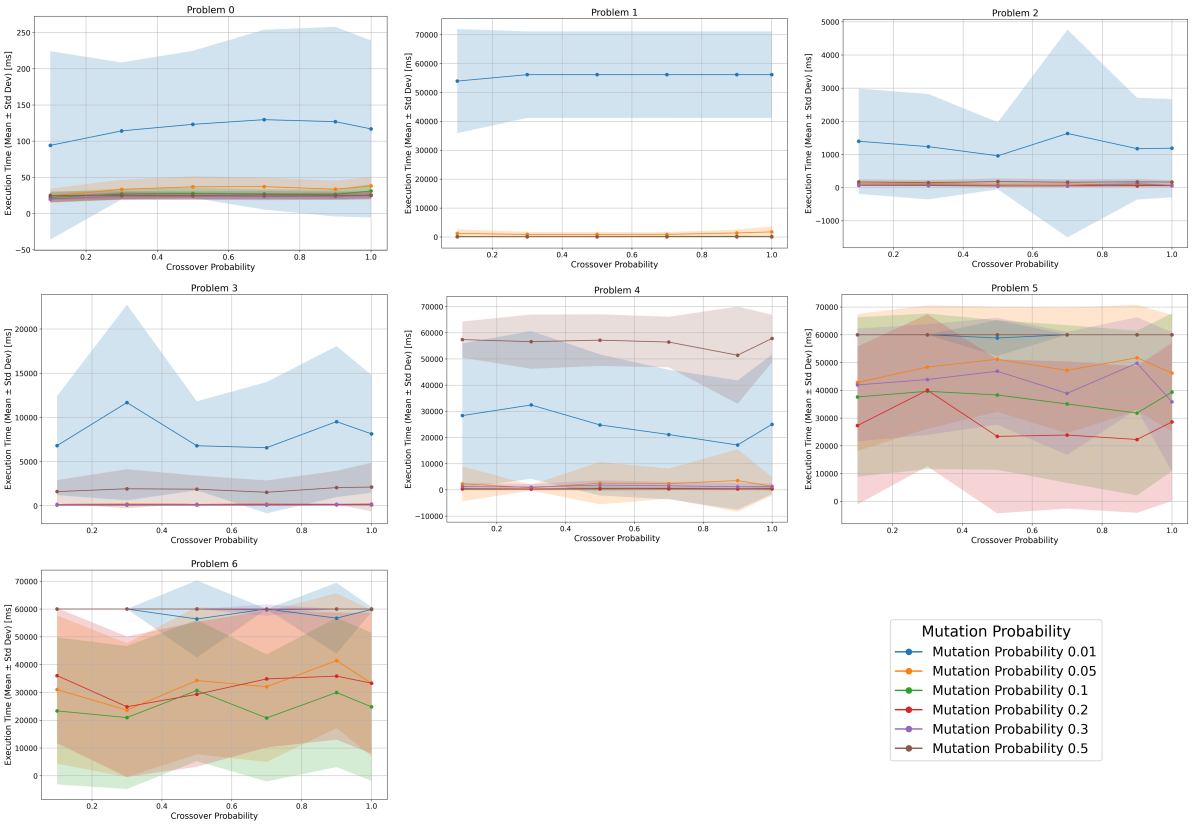

Figure 3: Mean and standard deviation of execution time with $N_e = *$.

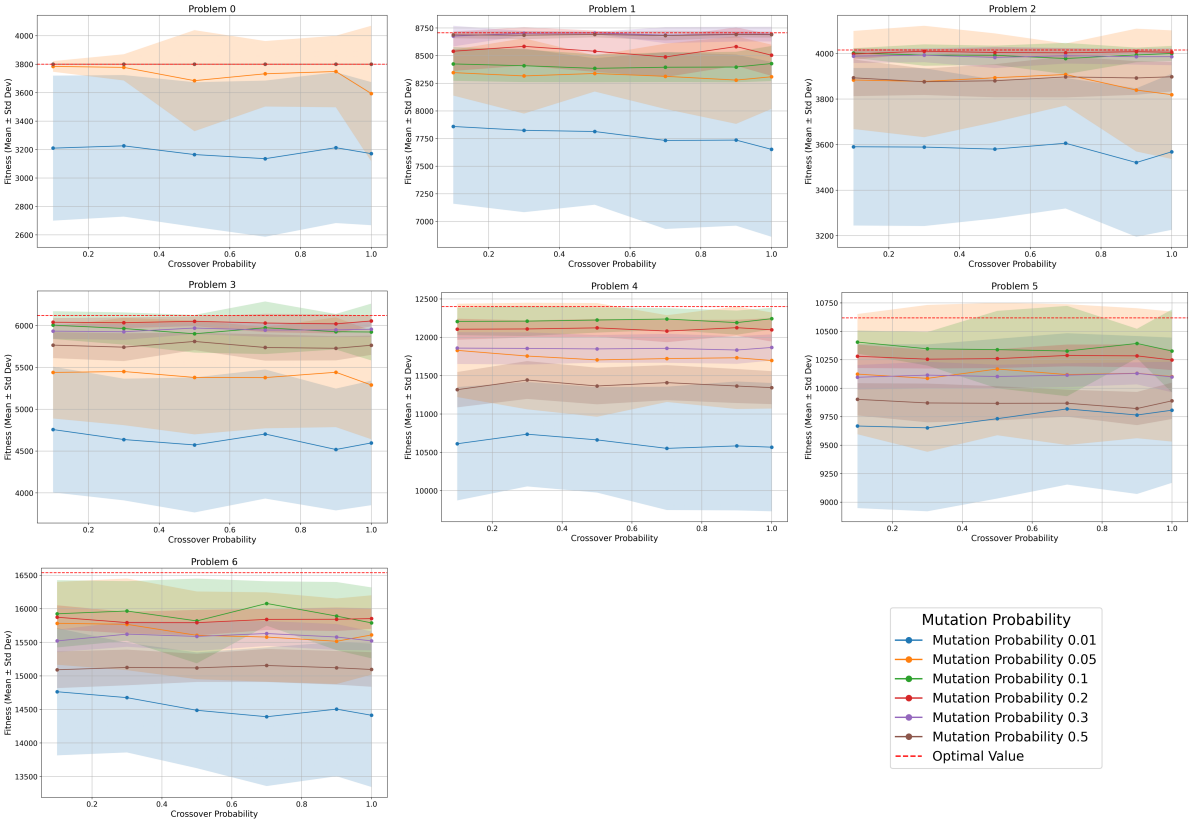

Figure 4: Mean and standard deviation of fitness with $N_e = 1000$.

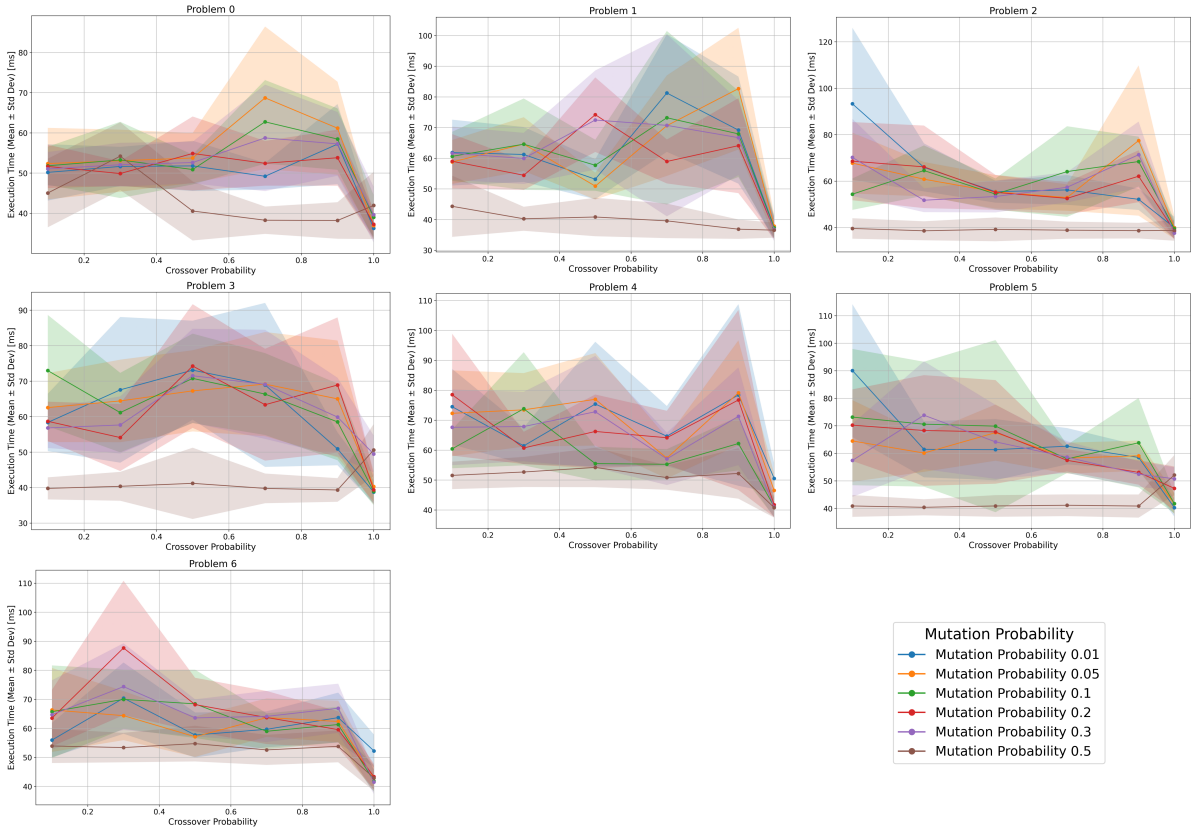

Figure 5: Mean and standard deviation of execution time with $N_e = 1000$.

B Result Tables for Performance Analysis

Table 2: Mean and standard deviation of fitness with $N_e = *$.

p_c	p_m	Problem 0	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6
0.1	0.01	3800.0 ± 0.0	8384.5 ± 125.8	4015.0 ± 0.0	6120.0 ± 0.0	12392.3 ± 15.4	10541.0 ± 58.3	16253.8 ± 442.3
0.1	0.05	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10596.0 ± 27.8	16517.0 ± 32.7
0.1	0.1	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10605.2 ± 16.1	16521.7 ± 31.0
0.1	0.2	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10611.1 ± 9.1	16524.5 ± 22.2
0.1	0.3	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10609.3 ± 9.7	16458.6 ± 22.5
0.1	0.5	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12363.2 ± 28.2	10476.2 ± 26.3	16189.0 ± 45.2
0.3	0.01	3800.0 ± 0.0	8360.7 ± 92.2	4015.0 ± 0.0	6120.0 ± 0.0	12373.5 ± 98.4	10518.5 ± 108.9	16208.0 ± 516.9
0.3	0.05	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10593.5 ± 27.4	16526.0 ± 27.0
0.3	0.1	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10604.2 ± 16.3	16520.5 ± 30.6
0.3	0.2	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10606.9 ± 9.9	16532.4 ± 8.0
0.3	0.3	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10608.7 ± 10.4	16466.1 ± 19.9
0.3	0.5	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12363.9 ± 25.4	10474.0 ± 21.0	16196.6 ± 56.5
0.5	0.01	3800.0 ± 0.0	8360.7 ± 92.2	4015.0 ± 0.0	6120.0 ± 0.0	12376.8 ± 98.4	10562.3 ± 45.4	16134.4 ± 567.8
0.5	0.05	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10589.1 ± 37.0	16473.4 ± 248.1
0.5	0.1	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10605.2 ± 14.8	16512.4 ± 36.0
0.5	0.2	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10612.0 ± 9.0	16527.4 ± 18.5
0.5	0.3	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10607.0 ± 10.6	16460.8 ± 26.9
0.5	0.5	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12367.1 ± 29.3	10472.2 ± 30.4	16205.2 ± 55.8
0.7	0.01	3800.0 ± 0.0	8360.7 ± 92.2	4015.0 ± 0.0	6120.0 ± 0.0	12394.2 ± 12.3	10548.2 ± 58.2	16038.1 ± 614.8
0.7	0.05	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10598.5 ± 26.4	16466.1 ± 247.8
0.7	0.1	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10608.2 ± 12.9	16522.8 ± 30.3
0.7	0.2	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10613.0 ± 7.9	16528.3 ± 14.6
0.7	0.3	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10609.3 ± 10.5	16469.2 ± 26.0
0.7	0.5	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12358.7 ± 28.5	10474.4 ± 30.2	16188.2 ± 38.7
0.9	0.01	3800.0 ± 0.0	8360.7 ± 92.2	4015.0 ± 0.0	6120.0 ± 0.0	12395.5 ± 13.1	10525.6 ± 102.5	16210.7 ± 490.2
0.9	0.05	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10590.2 ± 28.8	16501.9 ± 41.4
0.9	0.1	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10608.7 ± 10.5	16520.0 ± 27.9
0.9	0.2	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10613.4 ± 7.8	16526.1 ± 21.0
0.9	0.3	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10606.0 ± 10.9	16457.9 ± 19.5
0.9	0.5	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12370.0 ± 27.7	10476.2 ± 33.8	16183.6 ± 42.4
1.0	0.01	3800.0 ± 0.0	8360.7 ± 92.2	4015.0 ± 0.0	6120.0 ± 0.0	12375.2 ± 98.4	10544.4 ± 44.8	16190.3 ± 513.0
1.0	0.05	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10602.6 ± 17.0	16517.4 ± 30.7
1.0	0.1	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10604.7 ± 16.0	16517.9 ± 34.6
1.0	0.2	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10611.1 ± 9.1	16522.0 ± 26.0
1.0	0.3	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12400.0 ± 0.0	10610.8 ± 8.1	16465.5 ± 23.9
1.0	0.5	3800.0 ± 0.0	8706.1 ± 0.0	4015.0 ± 0.0	6120.0 ± 0.0	12358.7 ± 28.0	10482.0 ± 27.8	16188.9 ± 39.4

Table 3: Mean and standard deviation of execution time with $N_e = *$.

p_c	p_m	Problem 0	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6
0.1	0.01	94 ± 130	53903 ± 18001	1396 ± 1588	6799 ± 5596	28360 ± 27546	60000 ± 0	60000 ± 0
0.1	0.05	24 ± 10	1207 ± 1448	115 ± 62	163 ± 53	2293 ± 6614	42809 ± 24672	31010 ± 26677
0.1	0.1	22 ± 7	155 ± 110	75 ± 34	86 ± 27	477 ± 577	37592 ± 28704	23337 ± 26464
0.1	0.2	20 ± 5	51 ± 21	65 ± 20	82 ± 26	315 ± 254	27286 ± 28429	36037 ± 24338
0.1	0.3	20 ± 4	37 ± 12	75 ± 20	132 ± 54	1252 ± 1418	41922 ± 20311	60000 ± 0
0.1	0.5	25 ± 5	50 ± 16	170 ± 78	1599 ± 1301	57392 ± 6900	60000 ± 0	60000 ± 0
0.3	0.01	114 ± 94	56132 ± 14973	1234 ± 1590	11674 ± 11067	32421 ± 28216	60000 ± 0	60000 ± 0
0.3	0.05	33 ± 13	865 ± 865	131 ± 80	200 ± 74	796 ± 1115	48377 ± 22212	23610 ± 24049
0.3	0.1	27 ± 8	121 ± 82	80 ± 41	87 ± 23	302 ± 262	39633 ± 28018	20933 ± 25703
0.3	0.2	24 ± 5	37 ± 14	61 ± 18	84 ± 26	306 ± 264	40024 ± 27410	24825 ± 25273
0.3	0.3	24 ± 5	30 ± 8	74 ± 25	142 ± 56	1047 ± 1184	43856 ± 19908	60000 ± 0
0.3	0.5	25 ± 6	44 ± 13	147 ± 77	1918 ± 2211	56587 ± 10361	60000 ± 0	60000 ± 0
0.5	0.01	123 ± 102	56130 ± 14979	958 ± 1012	6785 ± 5029	24795 ± 26934	58842 ± 6448	56420 ± 13860
0.5	0.05	37 ± 15	862 ± 868	93 ± 60	179 ± 71	2566 ± 8068	51173 ± 18931	34263 ± 26518
0.5	0.1	28 ± 8	128 ± 61	59 ± 28	101 ± 35	540 ± 901	38303 ± 26925	30687 ± 25416
0.5	0.2	24 ± 5	36 ± 14	43 ± 13	87 ± 23	394 ± 295	23388 ± 27715	29306 ± 26071
0.5	0.3	24 ± 5	29 ± 8	55 ± 14	143 ± 74	1817 ± 1854	46864 ± 19223	60000 ± 0
0.5	0.5	25 ± 6	46 ± 18	181 ± 94	1870 ± 1539	57179 ± 9859	60000 ± 0	60000 ± 0
0.7	0.01	130 ± 124	56130 ± 14981	1631 ± 3131	6570 ± 7425	21119 ± 24744	60000 ± 0	60000 ± 0
0.7	0.05	37 ± 12	875 ± 780	104 ± 53	207 ± 76	2383 ± 5727	47160 ± 22583	32036 ± 27081
0.7	0.1	27 ± 6	172 ± 102	52 ± 25	96 ± 33	357 ± 371	35066 ± 28402	20794 ± 22847
0.7	0.2	24 ± 5	53 ± 22	47 ± 15	85 ± 25	512 ± 508	23868 ± 26506	34846 ± 24650
0.7	0.3	23 ± 5	37 ± 10	59 ± 19	162 ± 60	1552 ± 1452	38883 ± 22117	59637 ± 2024
0.7	0.5	26 ± 6	52 ± 19	162 ± 82	1523 ± 1335	56443 ± 9662	60000 ± 0	60000 ± 0
0.9	0.01	127 ± 131	56132 ± 14971	1172 ± 1534	9519 ± 8539	17124 ± 24658	60000 ± 0	56721 ± 12716
0.9	0.05	34 ± 12	1360 ± 1171	84 ± 37	164 ± 64	3548 ± 11941	51731 ± 19072	41442 ± 24242
0.9	0.1	26 ± 7	208 ± 144	62 ± 20	109 ± 28	326 ± 366	31789 ± 29639	29949 ± 26843
0.9	0.2	24 ± 5	56 ± 23	50 ± 12	100 ± 37	387 ± 356	22267 ± 26409	35855 ± 22864
0.9	0.3	24 ± 5	44 ± 13	113 ± 55	170 ± 55	1186 ± 1400	49760 ± 16562	60000 ± 0
0.9	0.5	25 ± 5	50 ± 16	172 ± 83	2051 ± 1893	51428 ± 18549	60000 ± 0	60000 ± 0
1.0	0.01	117 ± 122	56130 ± 14980	1187 ± 1481	8131 ± 6658	25027 ± 26590	60000 ± 0	59847 ± 851
1.0	0.05	38 ± 12	1743 ± 1903	81 ± 42	209 ± 75	1370 ± 3506	46195 ± 20958	33310 ± 26254
1.0	0.1	31 ± 9	154 ± 97	54 ± 18	98 ± 38	563 ± 822	39316 ± 28395	24799 ± 26657
1.0	0.2	26 ± 6	48 ± 22	53 ± 18	102 ± 40	351 ± 282	28575 ± 28471	33309 ± 25373
1.0	0.3	25 ± 5	29 ± 9	60 ± 17	162 ± 62	1277 ± 1141	35847 ± 24992	60000 ± 0
1.0	0.5	25 ± 6	41 ± 12	165 ± 58	2113 ± 2752	57816 ± 9073	60000 ± 0	60000 ± 0

Table 4: Mean and standard deviation of fitness with $N_e = 1000$.

p_c	p_m	Problem 0	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6
0.1	0.01	3209.7 ± 509.5	7858.6 ± 698.3	3590.3 ± 345.7	4756.6 ± 751.2	10611.6 ± 738.6	9667.9 ± 721.0	14763.1 ± 948.1
0.1	0.05	3783.9 ± 37.4	8345.9 ± 207.4	3883.5 ± 215.2	5439.2 ± 550.3	11828.9 ± 608.2	10123.7 ± 528.0	15782.4 ± 617.0
0.1	0.1	3800.0 ± 0.0	8424.4 ± 152.6	4001.9 ± 20.9	6004.7 ± 166.0	12206.1 ± 178.4	10405.2 ± 101.5	15924.4 ± 500.9
0.1	0.2	3800.0 ± 0.0	8539.3 ± 178.0	3997.7 ± 24.9	6039.2 ± 49.3	12104.5 ± 136.7	10280.4 ± 103.1	15873.2 ± 177.6
0.1	0.3	3800.0 ± 0.0	8676.3 ± 93.3	3988.2 ± 30.2	5932.7 ± 90.1	11858.5 ± 206.7	10096.0 ± 107.3	15520.1 ± 166.5
0.1	0.5	3800.0 ± 0.0	8686.9 ± 29.9	3893.1 ± 80.4	5765.6 ± 154.8	11316.5 ± 230.1	9902.1 ± 141.6	15090.3 ± 272.9
0.3	0.01	3225.8 ± 497.3	7824.0 ± 740.4	3588.9 ± 346.5	4637.4 ± 726.8	10736.3 ± 680.5	9651.7 ± 733.0	14674.8 ± 816.3
0.3	0.05	3777.4 ± 92.0	8316.0 ± 340.6	3876.9 ± 244.5	5450.2 ± 640.7	11755.0 ± 694.4	10086.5 ± 644.9	15767.6 ± 683.2
0.3	0.1	3800.0 ± 0.0	8408.4 ± 148.3	3992.9 ± 46.9	5964.0 ± 192.5	12210.0 ± 194.1	10346.1 ± 146.7	15966.4 ± 444.4
0.3	0.2	3800.0 ± 0.0	8583.4 ± 174.1	4009.8 ± 8.1	6033.4 ± 64.6	12108.1 ± 111.4	10254.7 ± 81.3	15793.4 ± 160.9
0.3	0.3	3800.0 ± 0.0	8701.9 ± 20.2	3992.4 ± 30.7	5925.5 ± 96.8	11854.0 ± 209.6	10115.0 ± 116.4	15621.4 ± 188.2
0.3	0.5	3800.0 ± 0.0	8684.5 ± 36.2	3875.8 ± 58.0	5741.3 ± 166.3	11442.7 ± 246.2	9870.3 ± 169.9	15124.7 ± 265.4
0.5	0.01	3164.5 ± 508.3	7813.1 ± 662.0	3580.0 ± 304.6	4572.7 ± 806.6	10662.1 ± 686.7	9732.2 ± 701.2	14486.3 ± 862.3
0.5	0.05	3683.9 ± 355.1	8338.1 ± 163.2	3893.4 ± 194.6	5379.5 ± 680.2	11704.7 ± 741.1	10168.6 ± 581.8	15603.4 ± 653.2
0.5	0.1	3800.0 ± 0.0	8384.5 ± 125.8	3992.1 ± 42.0	5901.1 ± 225.0	12224.4 ± 163.5	10338.9 ± 340.2	15817.8 ± 630.9
0.5	0.2	3800.0 ± 0.0	8539.4 ± 186.8	4004.2 ± 22.3	6050.3 ± 57.8	12122.1 ± 116.3	10259.5 ± 82.3	15792.2 ± 190.8
0.5	0.3	3800.0 ± 0.0	8704.3 ± 10.1	3982.6 ± 46.4	5969.8 ± 72.2	11847.9 ± 179.5	10102.6 ± 103.1	15587.4 ± 216.6
0.5	0.5	3800.0 ± 0.0	8691.1 ± 26.1	3880.2 ± 72.6	5809.5 ± 103.0	11364.5 ± 238.6	9867.2 ± 149.9	15118.5 ± 209.5
0.7	0.01	3135.5 ± 548.7	7732.3 ± 800.3	3605.8 ± 286.4	4703.4 ± 770.9	10551.5 ± 803.8	9818.1 ± 664.7	14390.4 ± 1032.9
0.7	0.05	3732.3 ± 230.1	8312.4 ± 295.4	3907.6 ± 136.0	5377.6 ± 605.9	11721.8 ± 565.4	10119.5 ± 618.3	15577.9 ± 666.0
0.7	0.1	3800.0 ± 0.0	8394.6 ± 134.2	3977.7 ± 68.7	5973.1 ± 314.0	12237.6 ± 142.2	10326.5 ± 396.5	16078.5 ± 330.8
0.7	0.2	3800.0 ± 0.0	8487.6 ± 181.6	4002.7 ± 20.3	6029.8 ± 110.2	12080.6 ± 144.9	10287.8 ± 95.4	15840.8 ± 157.4
0.7	0.3	3800.0 ± 0.0	8685.2 ± 72.3	3990.0 ± 34.2	5945.8 ± 80.9	11855.3 ± 188.5	10114.5 ± 104.2	15630.4 ± 185.2
0.7	0.5	3800.0 ± 0.0	8681.0 ± 43.4	3896.9 ± 89.8	5737.4 ± 157.3	11408.2 ± 227.8	9868.2 ± 118.3	15154.1 ± 246.0
0.9	0.01	3212.9 ± 529.6	7735.5 ± 774.1	3521.0 ± 325.8	4518.2 ± 727.7	10583.4 ± 840.0	9764.4 ± 693.4	14503.6 ± 999.6
0.9	0.05	3748.4 ± 251.5	8278.3 ± 395.8	3839.2 ± 269.6	5441.3 ± 655.3	11732.4 ± 668.0	10131.5 ± 570.7	15515.0 ± 637.2
0.9	0.1	3800.0 ± 0.0	8396.4 ± 138.0	3993.5 ± 32.0	5925.8 ± 211.9	12190.6 ± 157.4	10393.1 ± 128.7	15890.1 ± 507.4
0.9	0.2	3800.0 ± 0.0	8581.7 ± 173.7	4007.9 ± 9.0	6019.7 ± 114.7	12124.7 ± 103.3	10284.0 ± 99.9	15842.7 ± 174.6
0.9	0.3	3800.0 ± 0.0	8693.6 ± 66.3	3985.8 ± 34.6	5947.6 ± 84.2	11833.5 ± 179.4	10130.2 ± 96.3	15578.4 ± 192.4
0.9	0.5	3800.0 ± 0.0	8691.1 ± 27.4	3892.4 ± 73.9	5727.7 ± 144.2	11364.2 ± 221.4	9821.3 ± 145.5	15120.2 ± 250.6
1.0	0.01	3171.0 ± 502.8	7651.8 ± 790.6	3568.2 ± 342.6	4596.8 ± 743.6	10567.1 ± 835.9	9806.6 ± 637.8	14412.2 ± 1068.6
1.0	0.05	3593.5 ± 476.1	8308.3 ± 290.5	3819.0 ± 282.2	5289.4 ± 648.2	11696.8 ± 625.9	10102.3 ± 571.0	15609.3 ± 590.8
1.0	0.1	3800.0 ± 0.0	8428.0 ± 158.6	3999.7 ± 33.3	5922.3 ± 339.2	12241.8 ± 141.7	10326.5 ± 364.5	15789.0 ± 528.2
1.0	0.2	3800.0 ± 0.0	8503.6 ± 186.8	4005.3 ± 8.7	6054.4 ± 64.3	12096.8 ± 152.3	10247.8 ± 88.9	15853.1 ± 153.2
1.0	0.3	3800.0 ± 0.0	8694.2 ± 66.3	3986.0 ± 37.5	5956.1 ± 91.4	11866.6 ± 170.3	10099.0 ± 133.5	15520.8 ± 140.3
1.0	0.5	3800.0 ± 0.0	8689.9 ± 27.1	3897.6 ± 64.5	5764.0 ± 114.1	11343.1 ± 214.5	9889.0 ± 159.3	15094.5 ± 257.2

Table 5: Mean and standard deviation of execution time with $N_e = 1000$.

p_c	p_m	Problem 0	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6
0.1	0.01	50 ± 7	62 ± 11	93 ± 33	58 ± 8	74 ± 13	90 ± 24	56 ± 6
0.1	0.05	52 ± 9	59 ± 7	68 ± 13	63 ± 10	72 ± 14	64 ± 15	66 ± 14
0.1	0.1	52 ± 5	61 ± 8	54 ± 7	73 ± 16	60 ± 6	73 ± 25	66 ± 16
0.1	0.2	52 ± 5	59 ± 9	69 ± 17	59 ± 6	79 ± 20	70 ± 13	64 ± 10
0.1	0.3	51 ± 4	62 ± 9	70 ± 17	57 ± 6	68 ± 13	57 ± 13	65 ± 12
0.1	0.5	45 ± 9	44 ± 10	40 ± 4	40 ± 3	52 ± 5	41 ± 4	54 ± 6
0.3	0.01	52 ± 5	61 ± 9	66 ± 10	68 ± 21	61 ± 6	61 ± 10	70 ± 12
0.3	0.05	53 ± 8	65 ± 9	61 ± 7	64 ± 12	73 ± 12	60 ± 7	64 ± 8
0.3	0.1	53 ± 9	65 ± 15	65 ± 11	61 ± 11	74 ± 19	71 ± 23	70 ± 10
0.3	0.2	50 ± 3	54 ± 5	66 ± 18	54 ± 9	61 ± 5	68 ± 20	88 ± 23
0.3	0.3	52 ± 5	60 ± 8	52 ± 5	58 ± 8	68 ± 12	74 ± 20	74 ± 15
0.3	0.5	54 ± 8	40 ± 4	39 ± 4	40 ± 4	53 ± 5	40 ± 3	53 ± 5
0.5	0.01	52 ± 6	53 ± 7	55 ± 5	73 ± 14	75 ± 21	61 ± 11	58 ± 8
0.5	0.05	54 ± 6	51 ± 4	55 ± 7	67 ± 11	77 ± 15	68 ± 10	57 ± 7
0.5	0.1	51 ± 4	58 ± 8	54 ± 7	71 ± 13	55 ± 6	70 ± 31	68 ± 12
0.5	0.2	55 ± 9	74 ± 12	55 ± 8	74 ± 17	66 ± 12	68 ± 19	68 ± 9
0.5	0.3	53 ± 5	72 ± 16	53 ± 7	72 ± 13	73 ± 19	64 ± 13	64 ± 6
0.5	0.5	41 ± 7	41 ± 6	39 ± 5	41 ± 10	54 ± 7	41 ± 4	55 ± 6
0.7	0.01	49 ± 3	81 ± 19	56 ± 6	69 ± 23	65 ± 10	63 ± 7	60 ± 6
0.7	0.05	69 ± 18	71 ± 16	53 ± 6	69 ± 15	57 ± 7	58 ± 6	64 ± 7
0.7	0.1	63 ± 10	73 ± 28	64 ± 20	66 ± 12	55 ± 5	58 ± 5	59 ± 6
0.7	0.2	52 ± 6	59 ± 7	53 ± 7	63 ± 16	64 ± 9	57 ± 4	64 ± 9
0.7	0.3	59 ± 13	71 ± 30	57 ± 6	69 ± 15	57 ± 9	59 ± 6	64 ± 9
0.7	0.5	38 ± 3	40 ± 6	39 ± 4	40 ± 4	51 ± 4	41 ± 4	53 ± 5
0.9	0.01	57 ± 10	69 ± 17	52 ± 5	51 ± 5	78 ± 30	58 ± 4	64 ± 9
0.9	0.05	61 ± 12	83 ± 20	77 ± 32	65 ± 16	79 ± 18	59 ± 6	62 ± 7
0.9	0.1	58 ± 8	68 ± 14	68 ± 11	59 ± 11	62 ± 9	64 ± 16	61 ± 6
0.9	0.2	54 ± 7	64 ± 15	62 ± 11	69 ± 19	77 ± 30	53 ± 4	60 ± 6
0.9	0.3	57 ± 8	67 ± 12	71 ± 14	60 ± 11	71 ± 16	52 ± 5	67 ± 8
0.9	0.5	38 ± 5	37 ± 3	39 ± 3	39 ± 3	52 ± 8	41 ± 4	54 ± 5
1.0	0.01	36 ± 2	38 ± 4	40 ± 5	39 ± 3	51 ± 5	40 ± 3	52 ± 6
1.0	0.05	37 ± 3	38 ± 4	40 ± 5	40 ± 4	47 ± 6	42 ± 4	43 ± 4
1.0	0.1	39 ± 5	37 ± 4	39 ± 4	39 ± 4	41 ± 3	42 ± 4	42 ± 4
1.0	0.2	37 ± 4	37 ± 3	38 ± 4	39 ± 4	42 ± 4	47 ± 8	43 ± 4
1.0	0.3	40 ± 7	37 ± 4	38 ± 3	49 ± 5	41 ± 3	51 ± 4	41 ± 4
1.0	0.5	42 ± 8	37 ± 2	39 ± 4	51 ± 7	41 ± 3	52 ± 7	43 ± 4