Mutation Adaptive Genetic Algorithm for Constrained Capacity Planning Problems in Operations Research

FLORIN ALEXANDRU PRIBOI, University of Twente, The Netherlands

Having the best planning for business processes is essential for maximising profit and reducing production times. The problem arises when a large number of variables, for instance, the number of products, workers, locations, or production times, have to be taken into account, as such a combinatorial problem is computationally expensive. In the past, Genetic Algorithms (GA) have been proposed for solving these optimisation problems, and they have yielded great results. This study explores the application of GAs in Operations Research and breaks down each design consideration needed to adapt the Standard Genetic Algorithm to solve optimisation problems. With these requirements in mind, a new Mutation Adaptive Genetic Algorithm (MAGA) is proposed.

Additional Key Words and Phrases: Genetic Algorithms, Capacity Planning, Black-Box Optimization

1 INTRODUCTION

In today's competitive market, it is paramount for businesses to find the most efficient way of utilising their resources to maximise productivity, reduce operational costs, and maintain a competitive edge while meeting customer demands effectively [1]. When planning the resources necessary for certain capacity loads, multiple parameters have to be taken into account. Finding the best values for numerical parameters is known as optimisation and is a subfield of Operations Research (OR). Some variants of optimisation problems are Linear, Nonlinear, Integer or Mixed-Integer Programming. Being an NP-hard combinatorial problem, it is infeasible to calculate each combination of parameters for large solution spaces.

There have been multiple solutions proposed for solving these problems, like branch-and-bound [9] or cutting-plane generation [2]. Modern numerical solvers use a combination of multiple types of algorithms to arrive at the optimal solution [9]. The caveat is that they also require an extensive amount of time for convergence, especially for a large number of parameters, and they may not work on highly nonconvex, nonlinear problems.

Another proposed solution is the use of Genetic Algorithms (GA). They are a class of metaheuristic search algorithms used for exploring a large solution space. First introduced by John Holland in 1975 [4], GA mimics evolution to arrive at the best candidate solution. At the beginning of the program, a population of candidate solutions is initialised, each with random "genes". These genes represent the encoding of parameters in the solution space that need to be explored. Afterwards, each candidate solution – called a phenotype – is evaluated using a fitness function, and the best individuals get to "reproduce" in a process called crossover. Individuals swap parts of their genes, thus producing offspring. After crossover, random mutations of genes occur with a certain probability to introduce diversity and attempt to avoid the local minima. Afterwards, a new generation is formed. The process is then repeated until all the candidate solutions converge or after a set number of steps.

Although being used widely for rapidly generating high-quality solutions, GAs also have some drawbacks. Firstly, evaluating the phenotypes with the fitness function can be time-consuming. Doing this for each trial solution over hundreds of generations can be computationally expensive. Secondly, it is important to consider the constraints present in most optimisation problems when evaluating a phenotype. In addition, there is no guarantee of finding the optimal solution. Depending on the fitness landscape, GA can get stuck in a local optimum [6].

This study aims to explore how genetic algorithms can be used for automated capacity planning. The new Mutation Adaptive Genetic Algorithm (MAGA) is introduced, which builds upon the GSMR algorithm, alongside a novel use of the percentage-wise mutation operator. The new algorithm is tested on common optimisation benchmarks and compared to various algorithms from the literature in terms of objective value and convergence time. To achieve this, the following research questions have to be answered.

(1) RQ1: What are the key design considerations and adaptations required for a Genetic Algorithm to solve constrained capacity planning problems effectively?
(2) RQ2: How does MAGA compare to current Genetic Algorithms from the literature?

The paper is structured as follows: Section 2 presents an overview of current work on Genetic Algorithms (GAs). In Section 3, the requirements for a GA to effectively address capacity planning problems are discussed in detail. Section 4 offers a comprehensive explanation of the MAGA algorithm, while Section 5 presents the results of the experiments conducted. Following the results, Section 6 includes a theoretical analysis of MAGA, and Section 7 concludes the paper.

2 RELATED WORK

A thorough literature review must be done to answer the research questions proposed. There are already pieces of literature regarding the use of GA in capacity planning and similar problems, each of them proposing different additions to the basic algorithm detailed by Mr. Holland [4]. Extensive research was also done on modifications of the algorithm to try to mitigate the limitations mentioned above. In this section, an overview of the current knowledge will be presented.

In the 2019 paper by Jankauskas et al. [5], the authors proposed two modified GAS for capacity planning in medium- and large-scale capacity planning problems in biomanufacturing. These GAS were compared to Mixed Integer Linear Programming (MILP) solvers and yielded great results in terms of time required to converge to the

TScIT 43, July 4, 2025, Enschede, The Netherlands

optimal solution. The first algorithm proposed used another evolutionary algorithm called Particle Swarm Optimisation (PSO) [7] to find the best values for the mutation and crossover rates in a process called meta-optimisation. This meta-optimised GA was tested on a medium-sized problem with one production facility, converging 3.6 times faster than the MILP solver. The second algorithm added an additional modification to how the planning is realised. The full problem was split into 15 different overlapping sub-problems, which were independently solved by local GAS, and the best solutions were combined into a global schedule. This second variant was tested on a large-scale problem involving multiple facilities, and it converged to a value within 0.8 of the global optimum faster than its counterpart. These results are promising for the use of GA in capacity planning, especially for smaller-scale problems.

Another paper published by Yousefi et al. [14] demonstrates the use of GA in resource planning. This study focuses on an emergency department at a hospital in Brazil, where multiple parameters need to be optimised to decrease the average length of stay. The authors proposed a GA with a chaotic mutation operation, and after testing the solution in a real-life setting, it was shown to decrease the average length of stay by 14%. The chaotic mutation operator was introduced as a mitigation against the tendency of the GA to get stuck in a local maximum.

The study of genetic algorithms and their optimal convergence abilities is not limited to their use in capacity planning. Early work by Srinivas and Patnaik [13] has shown that implementing adaptive mutation rates based on an individual's fitness greatly increases performance. In a more recent study, Basak et al. [1] have proposed that a rank-based adaptive mutation outperforms the fitness-based adaptive mutation proposed by Srinivas and Patnaik. This rank-based mutation probability ensures that the best individual is not changed and that the distribution of the fitness function does not influence the probability.

A Cellular Automaton is a type of computational model used to simulate the behaviour of cells interacting with their neighbours based on a set of rules. This simulation is representative of how individuals choose partners in real life, so naturally, it was used to model selection in the study of Genetic Algorithms. In their paper, Salto et al. [11] explain how a structured population leads to a slower diffusion of genetic traits throughout the population, resulting in neighbourhood "niches" and a greater population diversity. First proposed by Whitley in 1993, the Cellular Genetic Algorithm (CGA) has been widely used as a mechanism to try to stop premature convergence.

In a study published in 2022 [8], Kumar et al. proposed the Group Elite Selection of Mutation rates (GESMR) algorithm. In their approach, a set of mutation rates is evolved alongside the candidate solutions. Each mutation rate is assigned to a subset of the population and is mutated based on the increase in fitness of the group. No crossover operation is used to isolate the effect of the mutation operation. The authors found that GESMR outperformed previous adaptive algorithms in multiple tasks, such as optimisation and neuroevolution for Image Classification and Reinforcement Learning. One algorithm that outperformed GESMR in Image Classification is 15MR [10], but failed to generalise in other tasks.

An important aspect to consider when optimising is handling constraints. Due to the random nature of the initialisation and operations of the GA, a phenotype has the chance of becoming infeasible. The most common way of addressing infeasibility is by adding a large penalty, but other methods are also available. Santos et al. [12] describe a model where each phenotype is sorted into subsets according to the deviation from the constraints. Each set, called a band, represents a range for the deviation, with a smaller band assignment meaning a smaller deviation. When selected for mating, first the band is checked, and the individual with the smaller band gets to reproduce. If two individuals share a band, only then is the fitness taken into account. The bands get reduced after each generation, guiding the search towards the feasible region of the solution space.

A simpler approach was designed by Deep et al. [3] and it involves introducing the deviation in the fitness function. If there are no feasible phenotypes in the population, the fitness then becomes the deviation from the constraints. This ensures that the phenotypes closer to the feasibility space get selected for reproduction. If there are any feasible phenotypes, the fitness of the infeasible individuals will become their deviation plus the worst fitness of the generation. With this approach, an infeasible phenotype with a larger deviation but with a very small fitness value does not get to reproduce, a problem that occurred with normal penalisation techniques.

3 REQUIREMENTS

To answer RQ1, this chapter summarises all the necessary adaptations of the SGA necessary for solving constraint capacity planning problems.

3.1 Problem Definition

Capacity planning problems can be defined as finding the best values of some decision variables $X \in \mathbb{R}^D$, as to minimise an objective function f, subject to some constraints g

$$\min_{X \in \mathbb{R}^D} \quad f(X) \tag{1}$$

subject to:

$$\begin{aligned} & g(X) < y, y \in R \\ & a < X < b, a, b \in R \\ & X \in \mathbb{R}^D \quad \text{or} \quad X \in \mathbb{Z}^D \quad \text{or} \quad X \in \{0,1\}^D \end{aligned} \tag{2}$$

3.2 Constraint handling

The first problem to be tackled is constraint handling. There are multiple ways researchers have tried to handle constraints, with the most popular being adding a penalty to the cost function. This approach does not account for the range of the fitness function, nor the degree of infeasibility. An individual who deviates greatly from the constraints will be penalised the same as an individual with a small deviation, and if the first one has a smaller fitness value, it will have more chances to reproduce. This does not guide the search towards the feasible region, and a better method has to be used.

As in the papers mentioned above [3] [12], one way of measuring infeasibility is the degree of deviation from the constraint. As a constraint is an equation, it consists of a left-hand side (LHS) and a right-hand side (RHS), which represent $g(x)$ and y respectively

(see Equation 2). The deviation becomes a measure of the sum of $|RHS - LHS|$ for all the constraints of the problem. Using this measure as a degree of infeasibility helps guide the search towards the feasible region.

Other methods of constraint handling are problem-specific and rely upon the encoding or the decoding of the problem. If the problem is encoded so that no infeasible individuals can be generated, or the decoding automatically solves the constraints, no handling is needed.

One important thing to mention is equality constraints. During the experiments, when testing problems with equality constraints, no method consistently produced viable solutions. This is due to the stochastic nature of the genetic operators. As the modifications take place randomly, equality constraints restrict the solution space in such a way that GA operators are not designed to handle.

3.3 Operators for different parameter types

An optimisation function can have either binary, integer, or a mix of these parameters. Each type of value can be mutated with different types of operators, with continuous variables having the most operators, whilst for binary ones, there is only flipping the bit. Each mutation operator, such as uniform, Gaussian, or proportional mutation, alters the phenotype in distinct ways and has specific advantages and disadvantages that should be taken into account when designing an algorithm. A GA for capacity planning should be capable of handling different data types.

3.4 Fine-Tuning and Adaptiveness

The optimum value of hyperparameters can change during the execution of the algorithm, depending on the position in the fitness landscape. At the beginning of the program, or when stuck in a local minimum, higher mutation rates can prove to be more beneficial, as they promote exploration. The inverse is true for the last iterations; a smaller mutation rate fine-tunes a solution that is close to the optimum. In order to help traverse rugged fitness landscapes, a GA needs to be able to adapt its operators based on feedback from the evolution.

3.5 Encoding

Independent of the genetic algorithm itself, one important aspect to mention is the importance of how the problem is encoded. How a problem is encoded directly affects the search space, which in turn directly affects performance. If there is a way of preventing infeasible phenotypes directly from the problem encoding, the algorithm does not need to first find the optimal space. This enables the optimisation process to focus immediately on identifying high-quality solutions, thereby accelerating convergence and improving overall performance.

4 ARCHITECTURE DESCRIPTION

In light of the requirements established above, this paper proposes the Mutation Adaptive Genetic Algorithm (MAGA), an algorithm that is equipped to adapt to multiple fitness landscapes due to its dual mutation operator. MAGA adapts its mutation operator based on the feedback received from the change in fitness, whilst maintaining a balance between exploration and exploitation. This section explains each design choice of the algorithm.

4.1 Mutation Operators

Being an extension of the GESMR [8] algorithm, MAGA uses two different mutation rate groups, one for each operator. In Additive Gaussian Mutation, the mutation rate σ is scaled by a small value ϵ drawn from the Standard Normal Distribution and adds this noise to the current value of the gene.

$$X_{t+1} = X_t + \sigma_G \cdot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 1) \tag{3}$$

The second operator used is Percentage-wise mutation, given by equation 2, which scales the current value of the gene by the mutation rate σ, which represents a percentage.

$$X_{t+1} = X_t \cdot (1 + \sigma_P) \tag{4}$$

One disadvantage of this operator is early starvation of mutation rates. If the initial mutation rates that produce a good change in fitness are positive, negative mutation rates will get starved out, with no possibility of being reintroduced. A simple and effective mechanism to solve this problem, which has been used in the past in maintaining population diversity, is a process called immigration. Here, a portion of the population, usually made up of the least fit individuals, is being reinitialised. If a negative mutation rate is more beneficial, it will quickly start dominating the other mutation rates

For integer and binary variables, a small modification is necessary. Binary variables use a normal mutation with a 20% chance of mutating. Small Integer variables are not susceptible to small modifications, so some stochastic option is needed to decide how to round the new value to the nearest integer. For both mutation operators, one of the following rounding functions, both with a probability of 50% is used:

$$X_{t+1} = \begin{cases} \lceil X_{t+1} \rceil & \text{with probability } 0.5 \\ \lfloor X_{t+1} \rfloor & \text{if case 1 fails} \end{cases} \tag{5}$$

These two mutation operators together are fit to explore the fitness landscape in a different but complementary manner. More details about the dual adaptation can be found in the discussion section. After each generation, the mutation rate groups are themselves sorted by the best fitness change. Truncation selection is used on the resulting lists, after which they are mutated according to their respective equations:

$$\sigma_G^{t+1} = \sigma_G^t \cdot \alpha^{\varepsilon}, \quad \varepsilon \sim \mathcal{U}(-1, 1) \tag{6}$$

$$\sigma_P^{t+1} = \sigma_P^t \cdot (1 + \varepsilon), \quad \varepsilon \sim \mathcal{U}(-0.2, 0.2) \tag{7}$$

Where α is a global meta-mutation rate fixed a priori.

4.2 Group Selection of Mutation Operators

For an initial population of N+1 candidate solutions, there are two groups of mutation rates, G and P, of equal size, with the property that (G + P)| N. At each generation, the candidate solutions are sorted according to fitness value. The fittest individuals are chosen to be

mutated via truncation selection. The fittest individual, also called the elite, is preserved and passed to the next generation unaltered, thus ensuring that no exacerbating mutations set back the algorithm.

After selection, the new generation undergoes mutation via 2 options: Gaussian or Percentage-wise. Initially, both mutation operators get assigned the same number of groups, but after several generations, the groups either increase or decrease based on a relative change in fitness produced by the operator. As there is a one-to-one mapping between a mutation rate group and a group of solutions, probabilistic sampling of the two operators becomes unfeasible. The algorithm uses a greedy strategy to select how many groups are assigned to each operator, maintaining this value within a certain bound, in order to allow some exploration during the later stages of the algorithm. When deciding which operator to increase, only the past few generations are considered, as information from earlier trials becomes irrelevant during the later stages.

4.3 Constraint Handling

The most natural constraint handling method for this algorithm is Deep's use of deviation ([3]). At each generation, the fitness score is influenced by whether there are feasible individuals in the population, thus allowing for quick adaptation of fitness relative to feasibility. The fitness function for an infeasible phenotype is thus given by the equation:

$$f(x) = \begin{cases} |g(x) - y|, & \text{if there are no feasible solutions} \\ \text{deviation} + f_{\text{worst}}, & \text{otherwise} \end{cases} \tag{8}$$

When the algorithm finds a feasible individual, the fitness value of the other phenotypes, which represented deviation up to this point, gets adjusted, thus placing them at the end of the rank. The suitable individual is now considered elite and gets selected for the next generation. The entire algorithm is described in the pseudocode in Algorithm 1.

Algorithm 1 MAGA – Mutation Adaptive Genetic Algorithm

```
Input: Population size N, number of mutation groups P, G
Output: Best solution found
Initialize population Pop of size N + 1, mutation groups P, G
Sort Pop based on fitness f(X) using Equation (8)
Apply truncation selection on Pop
while termination conditions not met do
    Mutate G groups using Equation (3)
    Mutate P groups using Equation (4)
    Apply Equation (5) for integer variables
    Sort Pop based on fitness f(X) using Equation (8)
    Apply truncation selection on Pop
    Identify the operator that produced the best fitness change
    Adjust P and G group sizes based on the result from Step 10
    Apply truncation selection on mutation rates P and G
    Mutate P and G according to Equations (6) and (7)
end while
return best individual in Pop
```

5 RESULTS

The algorithm has been tested on multiple benchmarks created to artificially simulate different fitness landscapes with 100 dimensions for the solution space.All algorithms have been run on a population size of 196 individuals for 300 generations, until they converged or until there was no improvement for 50 generations. More details about all the hyperparameters of the MAGA algorithm used in these experiments can be found in Appendix B. To answer RQ 2, the following algorithms have been chosen for comparison:

(1) Standard Genetic Algorithm (SGA) using Percentage-Wise mutation and Lagrange crossover
(2) Cellular GA with L5 Neighbourhood search using Percentage-Wise mutation and Lagrange crossover
(3) Rank-AGA using Percentage-Wise mutation and Lagrange crossover
(4) GESMR with 14 mutation groups
(5) GESMR-P, a variation of GESMR but only using Percentage-Wise mutation, with 14 groups

These algorithms were chosen as they possess different adaptability techniques, and GESMR-P was specifically designed to test the Percentage-wise mutation operator in isolation. The mutation and crossover types were chosen after multiple experimental runs to assess which operator was the most beneficial for these numerical optimisation problems.

Besides GESMR and GESMR-P, which do not use the mutation and crossover rate as a probabilistic decision, the other algorithms have a fixed mutation rate of 0.2 and a crossover rate of 0.8. GESMR, GESMR-P and MAGA do not use crossover. During testing, it was found that crossover hurts evolution in these mutation-centric algorithms, because the changes in fitness cannot be traced back to the mutation operator. Without a direct relation between mutation and relative fitness change, the algorithm is not able to select the most beneficial mutation rates.

In order to assess performance, the evaluation criteria are the best fitness value, based on 50 independent runs, and the average number of generations it has taken the algorithm to stop. For problems where the algorithms did achieve the global optimum, the differentiating factor is the number of times they converged. Due to the multiplicative nature of the percentage mutation, all results have been rounded up to 5 decimal points to avoid small biases induced by floating-point errors. For an algorithm to be considered superior, it should show a stastitically significant difference at the 95% confidence level.

As it can be seen from Figure 1, MAGA converges to the optimum on more problems compared to the other algorithms, achieving the global minimum 100% of the time on Griewank, Ackley, and Shifted Sphere, and 92% of the time on Easom. No algorithm converged to the global optimum on the Rosenbrock and Schwefel benchmarks. It achieves this performance on a smaller number of generations.

Running pairwise t-tests for the convergence count for all the problems, the only algorithm that showed a statistically significant difference at the 95% confidence level is Rank-AGA on the Easom benchmark, with a p-value of 0.042. Calculating the risk difference shows that MAGA converged 8% less. This can be attributed to the relatively flat fitness landscape of the function, where differences

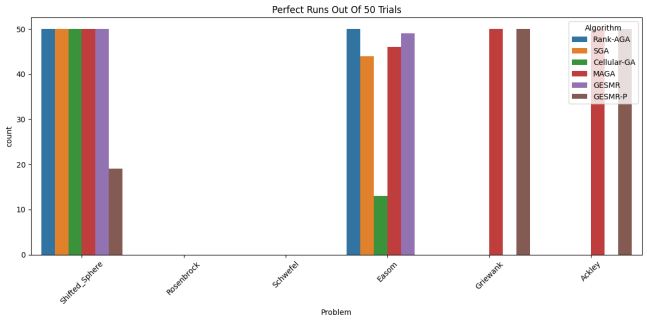

Figure 1. Number of Successful Global Optimisations (Unconstrained)

in objective value are relatively minimal from iteration to iteration, until the deep global optimum is reached. This, in turn, slows down the evolution of the mutation rates of MAGA and GESMR.

While each mutation operator excels individually on specific problem types, neither performs consistently well across all benchmarks. MAGA effectively combines their strengths, achieving competitive results on both classes of problems. This evidence indicates that the hybrid mutation strategy improves robustness and adaptability in diverse optimisation landscapes.

The same pattern can be observed in the functions where neither algorithm converged. MAGA achieves similar results as GESMR-P on Rosenbrock (Figure 2), and similar results as GESMR on Schwefel (Figure 3).

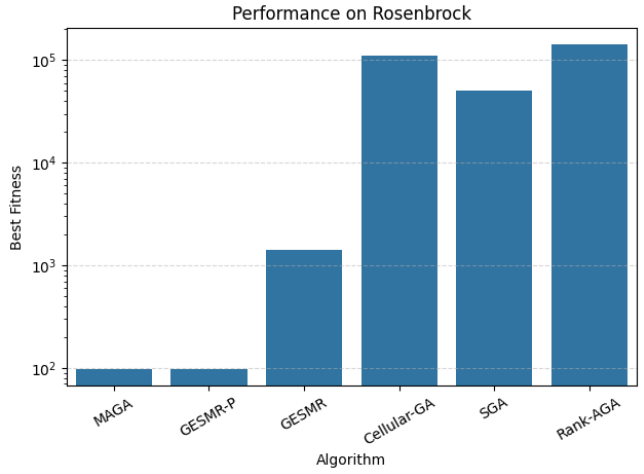

Figure 2. Best Fitness Value on Rosenbrock Benchmark

In terms of convergence speed, the use of multiple operators does not significantly slow down evolution. The average convergence generation can be seen in Figure 4. MAGA successfully uses both operators simultaneously, even when one of the operators performs worse in isolation. This implies that both operators help in converging, not that one operator takes over and the other just perturbs the solution, resulting in a weaker performance.

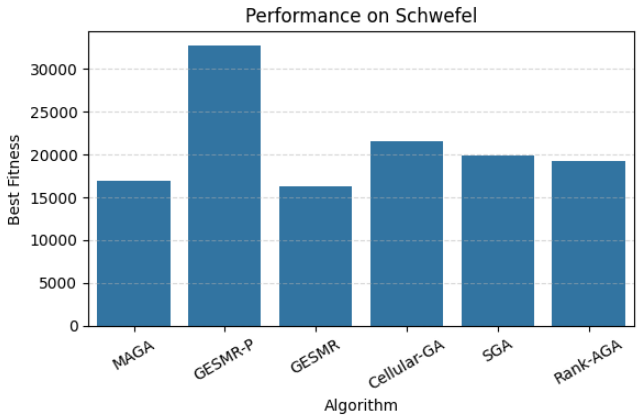

Figure 3. Best Fitness Value on Schwefel Benchmark

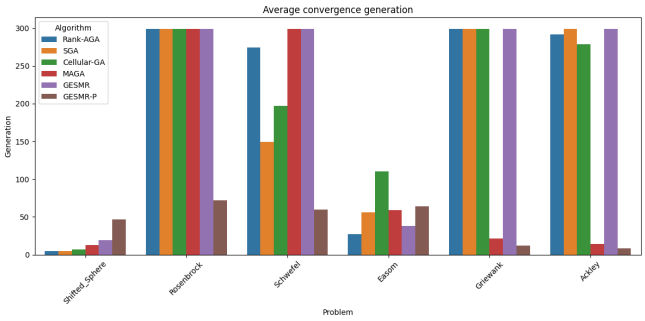

Figure 4. Average Termination Generation (Unconstrained)

To asses if the difference in convergence speed, a Kruskal-Willis and Dunn post-hoc tests were performed, as the convergence generations do not follow a normal distribution. After establishing that there is statistical difference between all the algorithms for each individual benchamrk, the Dunn test showed that MAGA was outperformed only by GESMR and Rank-AGA on the Easom benchmarmk. The results can be seen in appendix Appendix C. Although being outperformed on these tests, no other algorithm perfromed well across all the different benchmarks.

When it comes to constrained optimisation, all the algorithms were able to converge to a feasible individual in the 50 test runs. All the constrained benchamrks are the 2D variations. It is important to mention that all the algorithms use the same constraint handling method. Only Rank-AGA failed for a single optimisation run on the G08 function, but this can be attributed to unfavourable random initialisation. Consistent with previous observations, MAGA performs well across a diverse set of benchmarking problems, being the only algorithm that converged to the global optimum on all the optimization problems.

For the constrained benchamrks, Rank-AGA once more demonstrated better convergence abilities at the 95% confidence level on one of the benchmarks, namely Rosenbrock Constrained. In relation to convergence speed, MAGA is outperformed slightly on Mishra

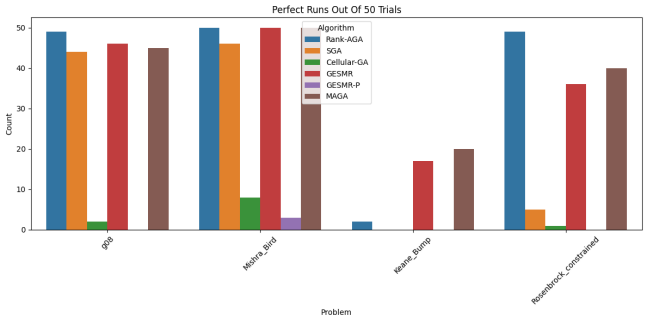

Figure 5. Number of Successful Global Optimisations (Constrained)

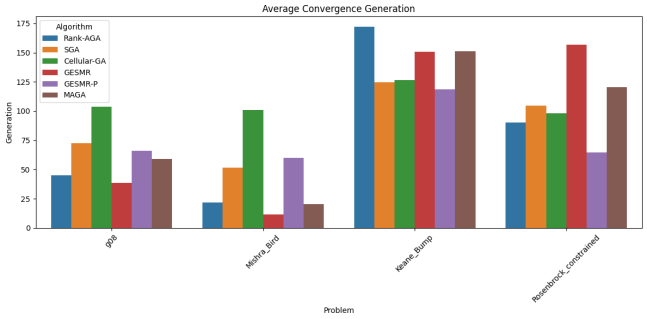

Figure 6. Average Termination Generation (Constrained)

Bird by GESMR. All the results of the Dunn tests can be seen in Appendix C.

In summary, the comparative analysis confirms that the proposed MAGA algorithm is the only algorithm that consistently produces good results across all the benchmarks, both constrained and unconstrained, without a great loss of convergence speed. The dual mutation operator proves to be effective in enhancing adaptability and robustness, maintaining the good results of the individual counterparts. These findings answer RQ 2, demonstrating that MAGA is a versatile alternative to traditional evolutionary approaches. MAGA represents a promising direction for future developments in evolutionary optimisation, especially in applications requiring generalisability.

6 THEORETICAL ANALYSIS

MAGA is designed to combine the logarithmical exploration properties of Percentage Mutation with the local search capabilities of Gaussian Mutation. The adaptive nature of the algorithm allows for wide exploration during the early stages and for fine-tuning at the end. In addition, the dual genetic operator maintains a better population diversity and gives the algorithm the possibility of choosing the best operator at each step of the optimisation process. The idea of the algorithm was that a solution would be scaled up or down by the percentage mutation, bringing it closer to the minimum, whilst the Gaussian mutation would adjust and fine-tune the solution at the end to be in line with the optimum, analogous to the slope and y-intercept in a regressor. This behaviour can be seen in Figure 7, which shows the number of groups assigned to each operator during a perfect optimisation of the Easom function.

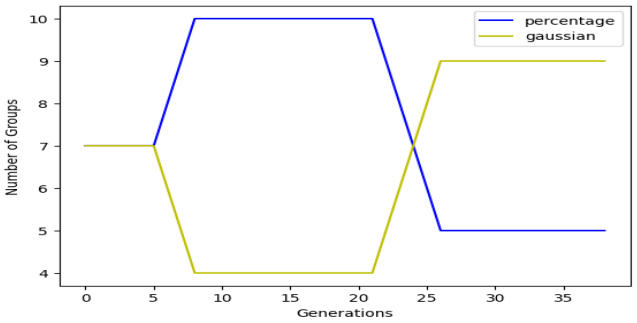

Figure 7. A Perfect Run on The Easom Benchmark

This was the expected behaviour, but it raised another question: why was the Percentage mutation better for the majority of the execution? During early experiments, GESMR had perfect results on a set of problems, while GESMR-P had poor results on the same set and vice versa. The initial expectation was that, for those problems, the respective operation would dominate during the execution. In practice, this is not the case. As can be seen from the result section, GESMR-P never came close to converging on the Easom function, but MAGA uses Percentage-wise mutation as the primary operator while converging to the global optimum. The same behaviour can be seen in this run for the Randomly-shifted Sphere function.

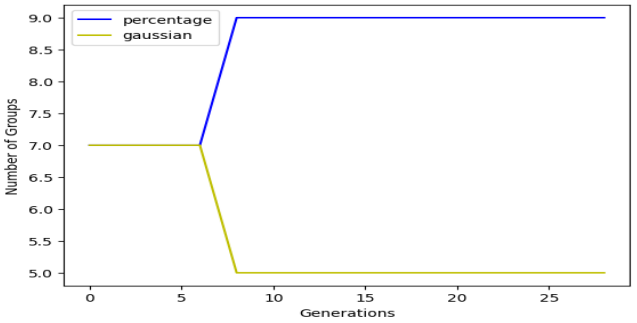

Figure 8. A Perfect Run on The Shifted-Sphere Benchmark

Percentage Mutation performed significantly worse on the Sphere benchmark, but on some runs of MAGA, it is the optimal operation (see Figure 8). It is important to note that this does not happen on all the runs; sometimes, one operator performs similarly or worse over the entire duration of optimisation, as shown in Figure 9.

One reason for this irregular pattern is population initialisation. As the population is initialised randomly, there is no telling a priori which operator would be beneficial at the beginning. In addition, there is no guarantee that a phenotype will be assigned a specific operator at a given iteration. Thus, the combined use of the mutations behaves as a different operator, but still achieves great results on the same benchmarks.

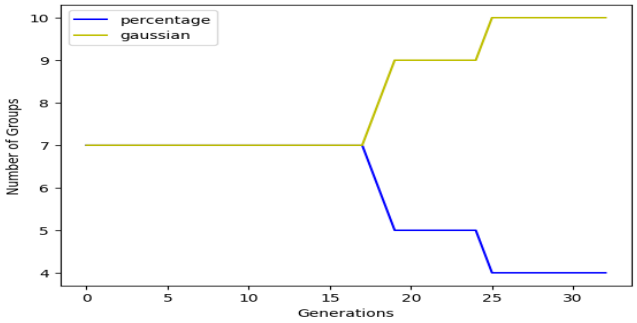

Figure 9. Another Perfect Run on The Easom Benchmark

A two-proportion z-test on the number of convergences on the Easom functions tells us that the difference between perfect runs is not statistically significant at the 95% confidence level. MAGA achieves the same results on the test functions where the individual operators perform great, but the combined use has a different effect on the population.

Unfortunately, there is no way to know a priori whether a genetic algorithm will perform well on a problem, and experiments need to be done to assess performance. According to the "No Free Lunch Theorem", two optimisers perform the same when averaged across all sets of problems. Since capacity planning problems can vary widely in terms of complexity and structure, no single optimisation algorithm can be considered universally superior. The results highlight the importance of empirical testing and adaptive strategies when selecting optimisation algorithms.

Since the experiments only focus on the use of two operators, further research has to be done in order to assess how more mutation functions combine together, and if a combination of multiple operators still preserves their individual results and increases generalizability, but also how this would affect convergence speed. In addition, the handling of integer variables is quite rudimentary, and further development is required to achieve better results.

7 CONCLUSION

To conclude, this paper presented the Mutation Adaptive Genetic Algorithm, a novel evolutionary optimisation algorithm specifically tailored to address constrained capacity planning problems in Operations Research. Building upon the strengths of previous algorithms such as GESMR, MAGA integrates a dual mutation strategy with a mechanism for adaptation based on fitness feedback. The design of MAGA was directly informed by a set of critical requirements identified in this study, specifically, the need for effective constraint handling, adaptive mutation, and support for multiple parameter types. All of these requirements were addressed in the proposed algorithm architecture.

MAGA proves it satisfies all the requirements listed for a capacity planning problem. Through extensive benchmarking on both constrained and unconstrained problems, MAGA demonstrated competitive performance compared to several established GA variants on a wider set of benchmarking functions. These results confirm that the hybrid mutation strategy not only maintains the strengths of its components but also introduces emergent behaviour beneficial for global optimisation.

These findings confirm that MAGA meets both research objectives: it fulfils the key architectural requirements for constrained optimisation (RQ1) and achieves superior performance compared to other genetic algorithms (RQ2). Additional work needs to be done to explore how the addition of multiple genetic operators influences generalizability and what the combined effect of the mutation operators has on population diversity.

REFERENCES

[1] Nils Arne Bakke and Roland Hellberg. 1993. The challenges of capacity planning. *International journal of production economics* 30 (1993), 243–264.
[2] Gérard Cornuéjols. 2008. Valid inequalities for mixed integer linear programs. *Mathematical programming* 112, 1 (2008), 3–44.
[3] Kusum Deep, Krishna Pratap Singh, Mitthan Lal Kansal, and Chander Mohan. 2009. A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Appl. Math. Comput.* 212, 2 (2009), 505–518.
[4] John H Holland. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press.
[5] Karolis Jankauskas, Lazaros G Papageorgiou, and Suzanne S Farid. 2019. Fast genetic algorithm approaches to solving discrete-time mixed integer linear programming problems of capacity planning and scheduling of biopharmaceutical manufacture. *Computers & Chemical Engineering* 121 (2019), 212–223.
[6] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications* 80 (2021), 8091–8126.
[7] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, Vol. 4. ieee, 1942–1948.
[8] Akarsh Kumar, Bo Liu, Risto Miikkulainen, and Peter Stone. 2022. Effective mutation rate adaptation through group elite selection. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 721–729.
[9] Ailsa H Land and Alison G Doig. 2009. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer, 105–132.
[10] Jeffrey T Linderoth, Andrea Lodi, et al. 2010. MILP software. *Wiley encyclopedia of operations research and management science* 5 (2010), 3239–3248.
[11] Carolina Salto and Enrique Alba. 2019. Cellular genetic algorithms: Understanding the behavior of using neighborhoods. *Applied Artificial Intelligence* 33, 10 (2019), 863–880.
[12] Maristela Oliveira Santos, Sadao Massago, and Bernardo Almada-Lobo. 2010. Infeasibility handling in genetic algorithm using nested domains for production planning. *Computers & operations research* 37, 6 (2010), 1113–1122.
[13] Mandavilli Srinivas and Lalit M Patnaik. 2002. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics* 24, 4 (2002), 656–667.
[14] Milad Yousefi, Moslem Yousefi, Ricardo Poley Martins Ferreira, Joong Hoon Kim, and Flavio S Fogliatto. 2018. Chaotic genetic algorithm and Adaboost ensemble metamodeling approach for optimum resource planning in emergency departments. *Artificial intelligence in medicine* 84 (2018), 23–33.

A AI ACKNOWLEDGMENTS

For the realisation of this paper, Grammarly and Quilbot were used to correct grammatical errors and misspellings. Additionally, ChatGPT was used to assist with LaTeX formatting and debugging during the implementation of the algorithms.

B ALGORITHM PARAMETERS AND DETAILS

As one iteration of the MAGA algorithm comprises multiple steps, numerous parameters that need to be initialised. At the beginning of the optimisation process, the initial population is randomly generated, with each gene being drawn from a uniform distribution with its upper and lower bounds.

The two mutation groups are initiated differently. Mutation rates for the Gaussian mutations are logarithmically spaced values ranging from $1e-3$ to $1e+3$. Percentage-wise mutation rates are linearly spaced values from -0.3 to 0.3. Initially, both mutation operators consist of 7 groups each.

A mutation rate is considered to be better if it produces a better average change in mutation in that specific generation. If a particular operator consistently performs better for five consecutive generations, the algorithm increases the number of groups using that operator, thereby reinforcing its influence.

Truncation selection is applied with a 25% truncation point, both for the solutions and the mutation rates. Half of the population with the Percentage-wise mutation rates are reintroduced every 5 generations.

C RESULTS OF DUNN TESTS

Table 1. Dunn post-hoc test for Ackley

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000
GESMR	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000
GESMR-P	0.0000	0.0000	1.0000	0.7479	0.0000	0.0000
MAGA	0.0000	0.0000	0.7479	1.0000	0.0000	0.0000
Rank-AGA	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000
SGA	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000

Table 2. Dunn post-hoc test for Easom

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	0.0000	0.0122	0.0000	0.0000	0.0000
GESMR	0.0000	1.0000	0.0000	0.0078	1.0000	0.0005
GESMR-P	0.0122	0.0000	1.0000	0.1314	0.0000	0.7860
MAGA	0.0000	0.0078	0.1314	1.0000	0.0000	1.0000
Rank-AGA	0.0000	1.0000	0.0000	0.0000	1.0000	0.0000
SGA	0.0000	0.0005	0.7860	1.0000	0.0000	1.0000

Table 3. Dunn post-hoc test for Griewank

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000
GESMR	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000
GESMR-P	0.0000	0.0000	1.0000	0.1160	0.0000	0.0000
MAGA	0.0000	0.0000	0.1160	1.0000	0.0000	0.0000
Rank-AGA	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000
SGA	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000

Table 4. Dunn post-hoc test for Rosenbrock

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000
GESMR	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000
GESMR-P	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000
MAGA	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000
Rank-AGA	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000
SGA	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000

Table 5. Dunn post-hoc test for Schwefel

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	0.0000	0.0000	0.0000	0.0002	0.7479
GESMR	0.0000	1.0000	0.0000	1.0000	0.4692	0.0000
GESMR-P	0.0000	0.0000	1.0000	0.0000	0.0000	0.0008
MAGA	0.0000	1.0000	0.0000	1.0000	0.4692	0.0000
Rank-AGA	0.0002	0.4692	0.0000	0.4692	1.0000	0.0000
SGA	0.7479	0.0000	0.0008	0.0000	0.0000	1.0000

Table 6. Dunn post-hoc test for Shifted Sphere

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	0.0010	0.0002	1.0000	0.1987	0.1156
GESMR	0.0010	1.0000	1.0000	0.1339	0.0000	0.0000
GESMR-P	0.0002	1.0000	1.0000	0.0487	0.0000	0.0000
MAGA	1.0000	0.1339	0.0487	1.0000	0.0017	0.0008
Rank-AGA	0.1987	0.0000	0.0000	0.0017	1.0000	1.0000
SGA	0.1156	0.0000	0.0000	0.0008	1.0000	1.0000

Table 7. Dunn post-hoc test for g08

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	0.0000	0.0058	0.0000	0.0000	0.0052
GESMR	0.0000	1.0000	0.0000	0.3400	0.0367	0.0000
GESMR-P	0.0058	0.0000	1.0000	0.0003	0.0065	1.0000
MAGA	0.0000	0.3400	0.0003	1.0000	1.0000	0.0003
Rank-AGA	0.0000	0.0367	0.0065	1.0000	1.0000	0.0071
SGA	0.0052	0.0000	1.0000	0.0003	0.0071	1.0000

Table 8. Dunn post-hoc test for Keane Bump

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	1.0000	0.4345	1.0000	0.0508	1.0000
GESMR	1.0000	1.0000	0.0591	1.0000	0.3850	1.0000
GESMR-P	0.4345	0.0591	1.0000	0.0585	0.0000	0.4727
MAGA	1.0000	1.0000	0.0585	1.0000	0.3884	1.0000
Rank-AGA	0.0508	0.3850	0.0000	0.3884	1.0000	0.0456
SGA	1.0000	1.0000	0.4727	1.0000	0.0456	1.0000

Table 9. Dunn post-hoc test for Mishra Bird

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	0.0000	0.0874	0.0000	0.0000	0.0021
GESMR	0.0000	1.0000	0.0000	0.0321	0.0022	0.0000
GESMR-P	0.0874	0.0000	1.0000	0.0000	0.0000	1.0000
MAGA	0.0000	0.0321	0.0000	1.0000	1.0000	0.0000
Rank-AGA	0.0000	0.0022	0.0000	1.0000	1.0000	0.0001
SGA	0.0021	0.0000	1.0000	0.0000	0.0001	1.0000

Table 10. Dunn post-hoc test for Rosenbrock Constrained

	Cellular-GA	GESMR	GESMR-P	MAGA	Rank-AGA	SGA
Cellular-GA	1.0000	1.0000	0.0003	0.7712	1.0000	1.0000
GESMR	1.0000	1.0000	0.0000	0.2493	1.0000	1.0000
GESMR-P	0.0003	0.0000	1.0000	0.3170	0.0193	0.0000
MAGA	0.7712	0.2493	0.3170	1.0000	1.0000	0.2790
Rank-AGA	1.0000	1.0000	0.0193	1.0000	1.0000	1.0000
SGA	1.0000	1.0000	0.0000	0.2790	1.0000	1.0000