KHAZAR UNIVERSITY

Faculty: Graduate School of Science, Art and Technology

Department: Computer Department

Major: Informatics

MASTER'S THESIS

Title: Hybrid Multi-Objective Genetic Algorithm for Emergency Response

Student: Ulkar Ahmadova

Supervisor: Dr. Saeed Saeedvand

June – 2022

Fövqəladə Hallar üçün Hibrid Çox Məqsədli Genetik Alqoritm

Ülkər Əhmədova

İnsan həyatından daha vacib heç nə yoxdur. Hər gün təcili yardım mərkəzlərinə müxtəlif vəziyyətlərlə bağlı çoxlu zənglər daxil olur. Bu vəziyyətlərin bəziləri risklidir, bəziləri isə yox. Tələbat yüksəkdir, lakin xilasetmə qruplarının sayı məhduddur. Tez və effektiv kömək göstərmək üçün cədvəl sürətli və dəqiq şəkildə yaradılmalıdır. Zamanın məhdud olduğunu nəzərə alsaq, ən yaxşı tənzimləməni tapmaq çox vaxt apara bilər. Bu problem əvəzində ən optimal yolu tapmaqla həll etmək olar.

İllər ərzində planetdəki bütün canlılar kəskin şəkildə dəyişdi. Bu dəyişikliklərə uyğunlaşmaq üçün insanlar, heyvanlar və hətta bitkilər hər nəsil təkamül etməyə davam etməlidirlər. Təbiətin mürəkkəbliyi neyron şəbəkələri, süni intellekt və s. kimi çoxsaylı alqoritm və texnikaları ilhamlandırıb. Təkamül alqoritmləri də onlardan biridir və optimallaşdırma üsullarına görə dəyişir. Genetik alqoritmlər insanların və heyvanların daim dəyişən mühitə uyğunlaşmasının təsirinə məruz qalmış ilk təkamül alqoritmlərindən biridir. Bioloji cəhətdən mükəmməl insan olmadığı kimi, bütün problemlərin mükəmməl həlli yoxdur, lakin genetik alqoritm ən optimalı tapmağa kömək edə bilər.

Genetik alqoritmin yaradılmasında ilk addım valideyn seçimidir. Ən yaxşı xüsusiyyətlərə malik ən uyğun olanları əldə etmək üçün seçilmiş genomların müxtəlif olması çox vacibdir. Seçim təsadüfi olaraq həyata keçirilir, bunun da öz üstünlükləri və mənfi cəhətləri var. Təqdim olunan alqoritm bir şəkildə seçim prosesini idarə etmək üçün Genetik Alqoritmlə birlikdə KNN alqoritmindən istifadə edir. Valideynlərdən biri təsadüfi seçildikdən sonra digər valideyn də təsadüfi, lakin fərqli sinifdən seçiləcək ki, bu da daha müxtəlif həllərin qoşalaşmasına səbəb olacaq. Həlllər məhlulun yaşına və onların uyğunluq funksiyasına əsasən qruplaşdırılır. Bu tədqiqat Fövqəladə Xilasetmə Qrupunun

Marşrutlaşdırma problemini həll etmək üçün edilir. Nəzərə alınacaq əsas məqsədlər xəstələrə çatmaq üçün vaxt, onların həyatını təhdid edən risk və xilas ola biləcək insanların sayıdır. Təklif olunan alqoritm 10 fərqli ssenari ilə sınaqdan keçirilib və daha sonra oxşar məsələ ilə bağlı mövcud araşdırma ilə müqayisə edilib. Hər iki alqoritmin müqayisəsinin nəticələri də müqayisə edilib və onlar göstərir ki, Hibrid Çoxməqsədli Genetik alqoritm yanaşması tələb olunan iterasiyaların sayını 12,28% azaltmışdır.

Hybrid Multi-Objective Genetic Algorithm for Emergency Response

Ulkar Ahmadova

There is nothing more important than a human's life. Every day, emergency centres receive lots of calls for various situations. Some of these situations are risky, some are not. The demand is high, but the number of rescue teams is limited. To provide help fast and effectively, the schedule should be created quickly and precisely. Finding the best arrangement may be very time-consuming, given the fact that the time is limited. This can be solved by finding the most optimal. Throughout years all of the living creatures on the planet changed drastically. To adjust to these changes people, animals and even plants have to keep evolving each generation. The complexity of the nature has inspired numerous algorithms and techniques, such as neural networks, artificial intelligence, etc. Evolutionary algorithms are also one of them and vary by methods of optimization. Genetic algorithms are one of the first introduced evolutionary algorithms, which have been influenced by the adaptation of humans and animals to the ever-changing environment. Just like there is no biologically perfect human, not all problems have perfect solutions, but genetic algorithm can help finding the most optimal.

The first step in creating a genetic algorithm is parent selection. It is crucial for the selected genomes to be diverse in order to get the best fitting ones with the best features. The selection is performed randomly, which has its own advantages and disadvantages. The presented algorithm uses KNN algorithm along with Genetic Algorithm in a hybrid way to control the selection process in a way. After one of the parents is randomly selected, the other parent will also be randomly selected but from a different class, which will cause more diverse solutions to be paired. The solutions will be grouped based on the age of the solution and their fitness function. This research is made to solve the Emergency Rescue Team Routing problem. The key objectives which will be considered are time to get to the patients, the risk that threatens their lives and the number of the people that can be saved. The proposed algorithm was tested with 10 different scenarios, and later will be compared to an existing research done regarding the similar issue. The results of the comparison of both of the algorithms were also compared and they demonstrate that the Hybrid Multi-Objective Genetic algorithm approach has decreased the number of required iterations to converge fitness value by 12.28%.

Table of Contents

1. Introduction

The purpose of the research is to find an optimal approach to provide emergency help to as many people as possible in the shortest time possible. When people's life is at stake, every second counts. This includes time for the arrangement of the tasks and locations between the rescue teams. The distance between the centre and the accident location, and the time it takes to reach there is one of the main objectives for the emergency service routing problem. Another aspect of the importance of effective team allocation is the cost of the operation. Sometimes, the teams are not equipped equally; some may have more devices which are needed to specific cases, while others have to utilize the basic equipment. The demand for emergency services stays high even with the rapidly evolving technology. Finding the best route considering all of the details of the situation can be very time consuming and even in some cases counter-effective, given the fact that there is a high possibility that the person needs to be immediately rescued. In some cases, when the number of accidents is very high, evaluating all of the existing combinations can take hours. This defeats the purpose of the emergency service, because this can cause serious delays.

Other important aspect of Emergency Service Routing problem is the seriousness of the accident. Needless to say, the more serious the situation is, the faster the rescue teams should arrive to eliminate the risk for the person's life or health.

The last aspect that will be mentioned is the number of people at the location. Naturally, the teams should first head to the places where the number of people that need urgent help is higher. This way, there will be more saved people without wasting time on the road, and the rescuers can help several people simultaneously. Instead of finding the best solution, with the disadvantage of time being spent on the searches, in this paper, the optimal solution is found. Numerous studies have

shown that genetic algorithms can greatly help with the optimization of the solution of the defined problem. Genetic algorithms produce solutions that can be very close to the best solution and require less time.

The contributions of this paper consist of the following points:

- Creating a Genetic Algorithm with implementation of KNN algorithm in the selection phase to regulate the chromosome selection to pair. This will insure some more diversity of the picked chromosomes. This step will reduce some randomness from the usual genetic algorithm.
- Creating a fitness function that will consider 4 objectives:
    - Time needed to reach the accident location from the emergency centre
    - Time needed to reach the accident location from another location
    - The risk for the people's life
    - The number of people at site
- Comparison of the proposed algorithm with an existing study. The study is related to Emergency Medical Service routing problem, and consists of 2 parts. For comparison only 1 part will be used, and it is the part where the routing problem is being optimized.

1.1. Genetic Algorithms

In the recent decade, evolutionary algorithms have become a popular optimization and search approach. Evolutionary Algorithms are a subcategory of Evolutionary Computations and are part of a group of current heuristic-based search methods. It becomes an effective way of problem solution for extensively used global optimization issues because to its flexible character and resilient behaviour acquired from Evolutionary Computation. It works well in a variety of high-complexity situations.

Evolutionary algorithms use iteration through generations with individuals evolving each iteration. In genetic algorithms the individuals are represented in a form of chromosomes. Each of the chromosomes consists of several genes which contain a piece of information within themselves [12]. These genes will be mixed and altered throughout the evolution process which will result in creating new chromosomes that will be named "**offspring chromosomes"**. Depending on the selection method chosen, the population consisting of parent chromosomes can either be fully or partially replaced by the newly created child chromosomes.
The main operations performed on the chromosomes each iteration are [9]:

1. Selection – the parent chromosomes are chosen
2. Crossover – result of pairing of the parent chromosomes
3. Mutation – altering a gene in the resulted chromosomes

1.1.1. Population Initialization

The first step in genetic algorithm is population initialization. The population will be the space of search of the defined problem. In that research, the initial population is randomly generated, limited by the following conditions:

1. All of the emergency locations will be included in the chromosomes as genes
2. None of the genes (locations) should be repeated in the chromosome
3. Each of the teams should have at least 1 location which they will be headed to

1.1.2. Selection

There are various **selection** methods, the main being Roulette wheel selection:

- Roulette wheel selection – going by the name of Fitness proportionate selection. All of the individuals for the next generation are chosen using the roulette wheel technique. In a genetic algorithm, it is a common selection

strategy. Each individual's relative fitness (ratio of individual fitness to overall fitness) is used to create a roulette wheel.

- Tournament selection - In a Genetic Algorithm, Tournament Selection is a Selection Strategy for picking the candidates the best fitting values out of the most recent generation. After that, the selected candidates are moved on to the next generation.
- Rank selection first rates the existing chromosomes, and then each chromosome's fitness is set on by that ranking. The worst will have fitness number one, the 2nd worst will have fitness number two, and so on, while the best will have fitness N which is the amount of chromosomes in population. Following that, all chromosomes have a possibility to be selected.
- Elitism selection – The goal is to sort the chromosomes in decreasing order of fitness. Afterwards perform the selection to each of the arranged set's two chromosomes. The Genetic Algorithm will then be used between weak and strong chromosomes in this manner. This indicates that no Genetic Algorithm can be used to distinguish between weak and strong chromosomes.
- Steady State Genetic Algorithm – or simply SSGA, is the type of selection in genetic algorithms where 2 randomly parents are selected and the resulting chromosomes take the place of the 2 chromosomes with the lowest fitness values. The key principle behind SSGA is that a significant proportion of chromosomes would be passed along to the following generation. Each generation, a few the ones with high fitness, chromosomes are chosen to create new offspring. Then certain chromosomes, with lowest fitness function, are withdrawn, and the new child gene is inserted instead. The rest of the population is passed on to future generations [7].

1.1.3. Crossover

After the parent chromosomes are chosen, the next operation is performed, which is **crossover**. The crossover operator, also called the crossover, is the main genetic operator by which genetic material is exchanged between individuals. It simulates the process of crossing individuals. A point within a chromosome is randomly determined at which both chromosomes divide into two parts and exchange them. This point is called the crossover point. There are 3 main classes of crossover [25]:

1.1.3.1. Standard crossover

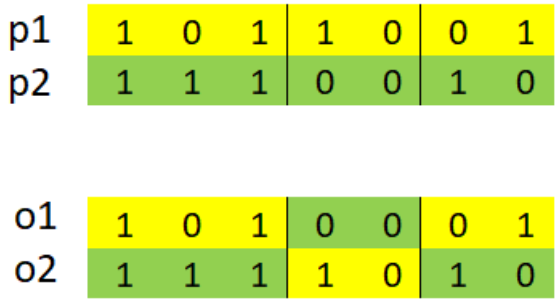
- 1-Point crossover – In this type or crossover operation, 1 point is randomly selected which will divide both of the parent to later combine them to create 2 new chromosomes. After the chromosomes are divided, the first fragment of the first parent is merged with the second fragment of the second parent and vice versa. The image below shows how the parent chromosomes divide and merge, where the yellow genes belong to the first parent and the green genes belong to the second parent.

p1	1	0	1	1	0	0	1
p2	1	1	1	0	0	1	0

o1	1	0	1	0	0	1	0
o2	1	1	1	1	0	0	1

- K-Point crossover – This type of crossover is similar to 1-point crossover. Here, instead of select 1 random point for the chromosomes to divide, there are k points. Similarly to 1-point crossover, there are 2 parent chromosomes that interchange their genes, but in k point crossover, 1 offspring can contain

more than one segment of each of the parent chromosomes. In the image below the yellow genes belong to the first parent and the green genes belong to the second parent.



- Shuffle crossover – This method is one of the fundamental crossover methods. Just like in 1-point crossover, a single crossover point is randomly chosen. However, before conducting the crossover, the genes in the parent chromosomes are randomly shuffled, and the switch is then made based on the new locations. The shuffle is the same for each of the parents. The positions are all reshuffled after the crossover. As a consequence, any positional bias is eliminated since the variables are shuffled randomly every time the crossover is performed. In practice, this strategy is comparable to the uniform crossover technique.

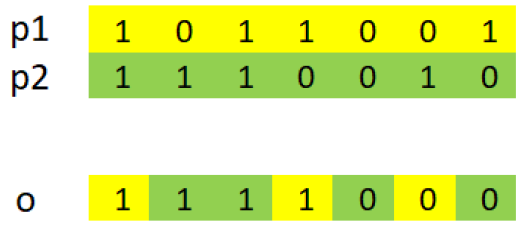
  The steps are listed as follows:

  1. Choose Shuffle points
  2. Shuffle the genes as defined by shuffle points
  3. Choose and perform 1-point crossover point
  4. Choose unshuffled points same as shuffled points
  5. Unshuffle the genes in the offspring chromosomes

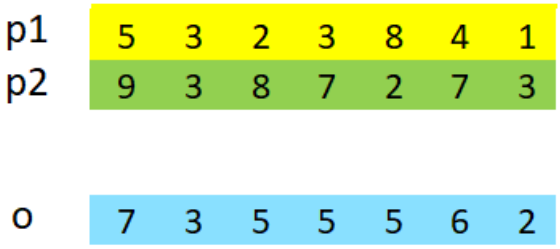
- Uniform crossover - This type of crossing is radically different from the previous types. Here, each gene of the offspring is generated by copying the

corresponding gene from the first or second parent, that is, each position is potentially a crossover point.

To do this, a binary crossover mask of the same length (with the same number of bits) as the chromosomes of the parents is randomly generated. The parity of the mask bit indicates the parent from which the child's gene is copied. For clarification, let's say that 1 corresponds to the first parent, and 0 to the second. For a mask [1 0 0 1 0 1 0], the uniform crossover will be done in the following way:



- Average crossover – this type of crossover is based on the value of the genes of the parent chromosomes. Here 2 parent chromosomes create only 1 offspring. The values of the genes from each of the parent are taken and the averages of these values are calculated. Later these calculated averages form the new offspring chromosome.



1.1.3.2. **Binary Crossovers:**

Random Respectful Crossover – Takes two parent chromosomes for crossover, and creates new offspring chromosomes using the parents' similarity vectors. It

first produces a similarity vector which includes the values of the parent if both genes of the parent chromosomes have the exact same value, else the similarity vector includes null value for that gene. Following the generation of the similarity vector, two offspring chromosomes are formed deriving from the similarity vector's values. If the similarity vector includes one, both children's genes are set to one, and if it contains nine, both children's genes are set to zero. Aside from that, if any gene in the similarity vector has a null value, the child gene is chosen using a uniform random real number.

- Masked crossover: To decide which bits of each parent chromosome are inherited by the child chromosome, the Masked Crossover operator uses a mask vector. The duplication of the parent chromosomes' bits is the initial phase. The first parent chromosome's bits are copied to the first offspring, and the second parent chromosome's bits are copied to the second offspring. In the next phase, the child chromosomes exchange bits at points where the parent's mask vectors are equal to 1, indicating domination of that parent at that position, and the other parent's mask vectors are equal to 0.
- Elitist crossover: The crossover process always comes before the selection process in a normal genetic algorithm. Both techniques are combined in the EX approach. The entire population is randomized at random in the first stage. Then, by crossover, two new vectors are formed from each consecutive pair of parental vectors. Two best vectors are selected from a 'family' and deployed as offspring in the following population. The traditional method of elitist selection, which is applied to the entire population, is often the cause of the algorithm's premature convergence. Applying elitist selection on a "family" level reduces this risk.

1.1.3.3. **Application dependant crossovers**

- Crossover for Traveling Salesman's Problems

  One of the main conditions of Traveling Salesman's Problem is that the cities should not be repeated and all of the cities should be visited. When performing Standard crossovers on these problems, in most cases the genes will be repeated. This problem can be solved by using crossover operators, specifically designed for TSP problems. There are a number of these crossovers: Order-Based Crossover (OBX), Modified-Order Crossover (MOC), etc.

- Other crossover application [25]:

  Some other problems that require special crossover operators are: Object classification problems, Crossover for Sudoku problem, Crossover for Graph Colouring Problem (for Parallel GA), etc.

1.2.3. Mutation

Mutation is a genetic operator which is used to keep genetic variety in a population of genetic algorithm chromosomes from one generation to the following. Without mutation, the Genetic algorithm may be stuck with the same set of solutions due to it simply exchanging some parts. Changing one of the genes may affect the fitness value greatly. There are several types of Mutation operators:

- Bit flip [22]: Bit-flip mutation is performed on binary represented genes. In this type of mutation, 1 or more genes are selected and the values are flipped, if the value of the gene was originally 1, then it will be changed to 0, and vice versa. The example of the bit-flip mutation operation is presented below:

  1 0 **1** 1 0 **1** **0** 1 1 → 1 0 0 **1** 1 **0** **1** 1 1

In this example, the red ones are the selected genes for bit flip mutation operation.

- Boundary – this mutation operator is used with genes with float values. The boundary mutation randomly chooses one gene and then replaces the value with either upper or lower value.
- Uniform – uniform mutation operation is used with both integer and float type genes. First a random gene is selected and then it is replaced by a random number from the range defined by the user.
- Swap – swap mutation is especially useful with problems like TSP. As mentioned before, the cities in TSP problem cannot be repeated in one solution. In this case one of the best solutions will be swap mutation. The principle of this mutation operator is choosing 2 random genes and swapping them. So, if the original solution is 1, 2, 3, 4, 5, the mutation result can be: 1, 5, 3, 4, 2.
- Scramble Mutation – Scramble mutation is also one of the possible mutation types for TSP problem. A portion of the chromosome is selected and then displaced. It is very similar to Swap mutation described above, but there are 2 key differences: First one is the number of genes affected. Swap mutation takes 2 genes, whereas scramble mutation can take more than 2. The second difference is that in Scramble mutation, the genes which are being affected are consecutive.
- Inversion mutation – Inversion mutation operator also makes changes in a portion of the chromosome, but in this case, the genes are being inverted. For example, if the original chromosome is represented as [1, 2, 3, 4, 5], the Inversion mutation result can be [1, 2, 5, 4, 3], where the last 3 genes were selected for the mutation to be performed.

1.3. KNN Algorithm

Machine learning (ML) is the science of computer systems which is able to learn from data and experience to upgrade themselves automatically. It is closely related to Artificial Intelligence. Machine learning algorithms generate a training data-based model to create predictions or decisions without forcing to be specifically programmed to do so. ML algorithms are implemented in a broad variety of fields, some of them being email filtering, fraud detection, medicine, marketing, targeted advertisement, etc.

ML is tightly linked to computational statistics that concentrates on generating predictions with computers; nevertheless, statistical learning is not all about machine learning. The science of ML aids from the studies of mathematical optimization due to the fact that it provides techniques, concept, and fields of application. There are 3 learning methods in ML: Supervised Learning, Unsupervised learning and Reinforced Learning.

Supervised learning is the most used and researched kind of ML since it is easier to train a device that has chosen data. This learning method uses labelled data, which is called training data, to determine the labels of the test data, which is the part of the data that needs to be classified or predicted. Based on what one wants to forecast, supervised learning may be used to unravel 2 sorts of problems: a regression problem and a classification problem.

1.3.1. Regression task:

Regression is a statistical method used in business, marketing, and other areas [14] to determine the strength and type of a link concerning a dependent variable (usually represented by Y) and a collection of other variables (well-known as independent variables). Regression may be used by investment and financial

managers to evaluate assets and analyse relationships between elements such as commodity prices and the stocks of firms that deal in those commodities. When attempting to forecast continuous data, such as the cost of a house or the temperature outdoors in degrees, regression should be used. Because the value can be any number with no constraints, this task type does not have a defined value limit.

Linear regression models, in particular, demonstrate how one or more explanatory factors may explain a portion of the natural individual-to-individual variance in a continuous response variable. In my research I found it more suitable to proceed with classification task method.

1.3.2. **Classification task:**

The Classification method is a Supervised Learning strategy that determines the category of new observations using training data. The process of system learning from a data collection or observed data and then categorizing freshly submitted observations into one of several groups or groupings is known as classification. True or False, Spam or Not Spam, 1 or 0, and so forth. Classes are sometimes known as labels, or categories. Different from regression, this method generates a class in contrast of a value, like "Digit or Letter," "White or Black", “Normal or Suspicious”. Since the Classification approach is a supervised learning method, it makes a use of labelled input data, that suggests that it contains both input and output. There are several kinds of classification algorithms, such as:

- **Logistic Regression**

  Logistic Regression is a ML classification algorithm which employs 1 or more independent variables in order to produce an output [11]. A binary variable is needed to assess the result, and that leads to a conclusion that there are only 2 potential results. The resolution of logistic regression is to determine the most

suitable fit between a dependent variable and a collection of independent variables. Logistic regression is used to describe the way a group of independent aspects has an effect on the outcome of the dependent variable.

- **Stochastic Gradient Descent**

Stochastic Gradient Descent is an extremely effective and straightforward method for fitting linear models. When the sample data is vast, Stochastic Gradient Descent is especially beneficial. Various loss functions and penalties are supported for categorization. Computing the result from every training data point and computing the update instantly is referred to as stochastic gradient descent.

- **Naive Bayes Classifier**

Naive Bayes Classifier is a classification method founded on Bayes' theorem [26], that presumes the predictors are independent. Even though the qualities are dependent on each other, every one adds to the prospect individually. This model is easy to build and is particularly good for big datasets. The classifier uses just a little amount of training data to compute the needed parameters.

- **K-nearest Neighbours Classifier**

KNN is one of the essential Supervised Learning techniques. It is based on the fact that data points, which are located close to each other, should share enough characteristics to be considered being related to the same class [6]. In the image below, we can see a graph, with the train data, that is used to train the machine. Each of the colours represents one of the available classes. For example, let’s assume that the green dots are representing the birds’ category in animal distinction algorithm, the red ones represent mammals and the blue ones represent fish. The data points are illustrated in Figure 1. In the graph we can see that the dots related to each of the classes are located closely to each other.

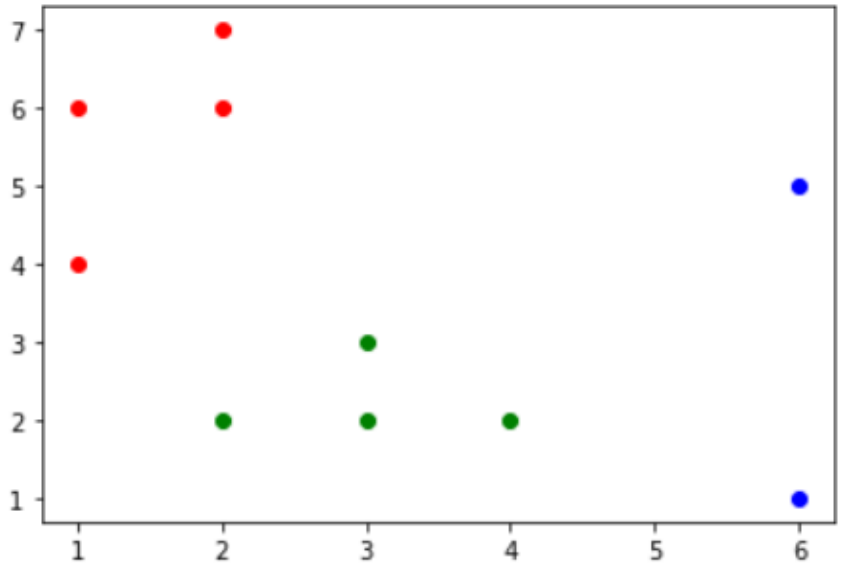


Figure 1. Train data for the classification problem

Let's assume new data was added, that needs to be categorized. There are 5 new points. The points are added to the previous graph as black dots, and are illustrated in Figure 2.

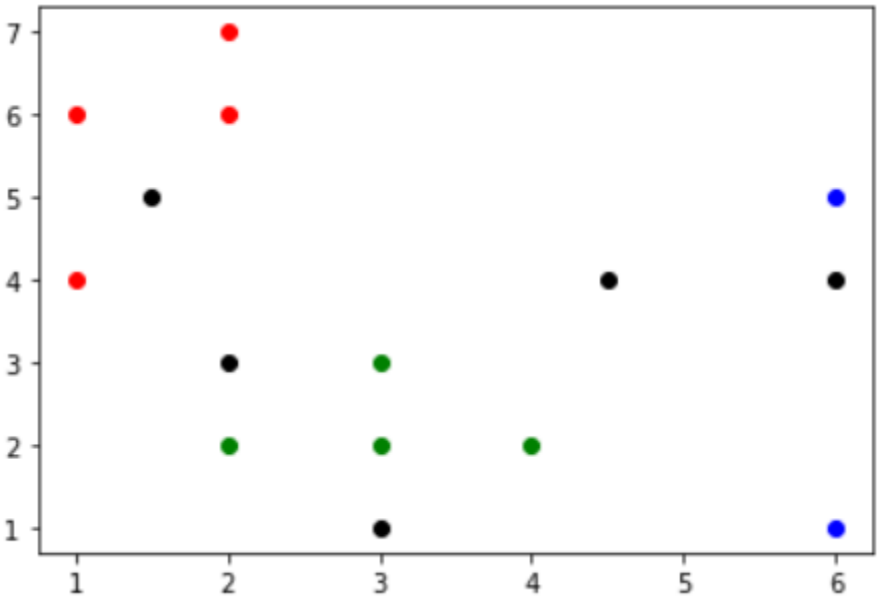


Figure 2. Train data and Test data (black)

To classify each of these points using KNN algorithm, we need to find the distance between each of the new points and the old points. The newly added data is called

test data. First, we need to find the distance between the points from test data and every point of train data. To find the distance, Euclidean distance formula is being used. The formula is shown in Eq.1:

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Eq 1. Euclidean distance

After finding the distance between each of the points, we need to determine the number of the nearest neighbours, based on which the point from test data will be classified. For this example, let's assume the k number, the number of closest neighbours will be 3. The closest train data points to the first point are 1, 2, and 4. Their classes are red, red, red, so the class of the 1st point is red. Similarly, the closest train data points to the second point are 2, 5, and 7. Their classes are red, green and green, so the class of the 2nd point is green. The closest train data points to the third point are 7, 8, and 10. Their classes are green, green and blue, so the class of the 3rd point is blue. The closest train data points to the fourth point are 5, 6, and 8. Their classes are green, green and green, so the class of the 4th point is green. The closest train data points to the fifth point are 8, 9 and 10. Their classes are green, blue and blue, so the class of the 5th point is blue. To clearly see the principle of the algorithm, the train data is now updated and includes the recently added test points. All of the points are shown below in Figure 3, with the colours being distributed respectively. As we can see, the newly added data points are close to their classes.

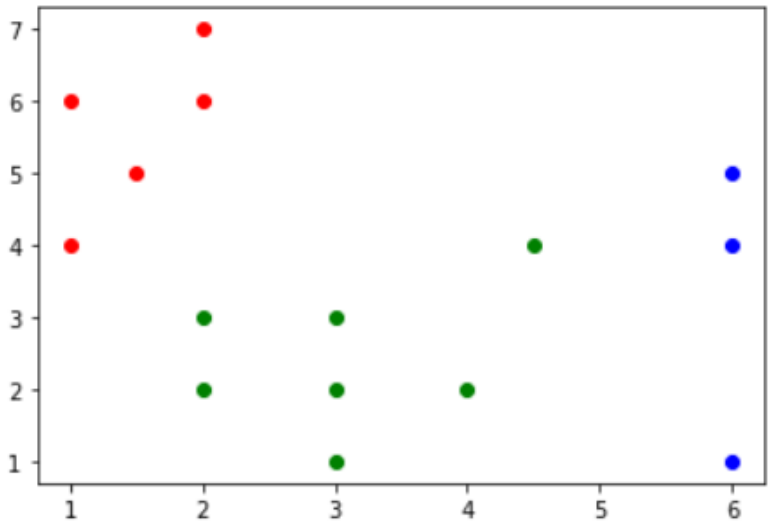


Figure 3. Test data is classified

2. Literature Review

In this section we discuss the existing solutions to emergency and rescue optimal route problems:

The first papers trying to solve this problem is presented by Bo Ai, Benshuai Li, Song Gao, Jiangling Xu, Hengshuai Shang [3].The most crucial aspect of maritime search and rescue (SAR) operations is making resolutions. The way to swiftly react to incidents and build an emergency response plan is a significant aspect determining efficiency and success rate while making maritime SAR decisions. The majority of marine SAR emergency response plans are created using an amalgamation of drift prediction models and SAR expertise. SAR resource scheduling and task assignment are both lacking. The main goal of this research is to investigate the feasibility of formulating maritime SAR emergency response plans utilizing an intelligent decision-making algorithm in order to achieve more scientific outcomes. The key technologies involved in developing marine SAR emergency response plans are discussed in depth in this study, and the maritime SAR decision problem is broken down into three sub-problems: SAR area determination, SAR resource scheduling, and SAR task assignment.

The following are the paper's main contributions:

1. The optimal sea search hypothesis has been enhanced.

   By introducing the POR, the notion of POSSAR is proposed and employed as the purposeful function of the resource scheduling model. That model prioritizes life-saving assistance, which is more reconcilable with actual SAR operations, and the resulting resource allocation plan is more logical.
2. GSAA solves the resource scheduling model, that can discover the global optimal solution while maintaining search efficiency and avoiding falling into the local optimal solution.
3. A novel regional job assignment method is being developed based on space-time properties. The algorithm generates a plan that covers the whole search region while taking task priority into account and avoiding overlapping task areas. This not only minimizes the likelihood of missing targets and enhances SAR efficiency, but it also cuts down on redundant searches, saving time and money in the process. Furthermore, the system executes phased search job planning, which increases the synergy between the SAR units.
4. The key difficulties in every section of the maritime SAR decision are puzzled out using various intelligence algorithms, and a full and optimal decision-making scheme is obtained quickly and intelligently, which not only reduces accident response time but increases SAR efficiency, too.

The decision algorithm for the SAR emergency response plan suggested in this research displays evident optimization through example verification, and can serve as a model for future marine SAR emergency response plans.

It will be looked upon into some of the unsolved decision-making difficulties in the future (for example, dynamic adjustment of decision-making programs, collaborative SAR of aircraft and ships, and so on). There are two key ideas at the

moment: using Adversarial Networks to optimize SAR decisions by imitating effective SAR operations; using Reinforcement Learning to improve SAR decisions. Adapting to realize interaction between agents (SAR units) and their surroundings (SAR environment), resulting in improved agent behaviour.

The second study was proposed by Ritu Pal, Manu Srivastava, Sudha Rani, Neeraj Kumar [19]. Reporting a flood disaster is crucial for potential victims and rescuers alike. Android applications are assisting flood survivors and rescuers in determining the exact location of the flood. Genetic algorithms may play an essential role in route optimization by providing precision in localization and route selection. MyDisasterDroid is a smartphone application that uses the same type of concept for disaster reporting (MDD). This application assists in providing accurate disaster location information to relief workers and rescuers; however it can be improved by correcting its methods. After pointing out certain typical flaws, this paper offers suggestions for improving it. This research also suggests that a parameter be included for a better route optimization approach. Slop has been treated as a parameter with distance to discover an initial solution in the path to the closest reservoir. Researchers employ geolocation as an initial input, with the best path being the intended outcome.

This application uses a genetic algorithm with a specific formula to discover the best path between two points. Researchers have used the concept of the travelling salesman problem to determine a course for water flow in order to avoid floods or large amounts of stored water at a location. In the event of a flood, the water does not return to its original location. The travelling salesman problem, in which geographic locations exhibit city coordinates and water represents the travelling salesman, is analogous to determining the optimal way to change the flow of water along distinct geographic locations. In this scenario, the TSP provides the shortest path to the city's empty pond or reservoir as a destination. With the use of GA,

these different paths can be used to select the best ones. After receiving location as an input, this algorithm generates the initial solution. MDD is based on the concept of geo-location for providing inputs to the genetic algorithm. It locates points of interest utilizing an MDD-installed application that sends the location to MDD through text or SMS. Distances between sites are computed using TSP. MDD employs a flow diagram but lacks the ability to detect a inefficacious route in a disaster, which is a real possibility. Failed disaster initialization routes are a major issue for every disaster. MDD for Android uses Google Maps to display its MapView feature. MapView offers a variety of map views, including satellite, street, and traffic views. Because Google Maps is unable to give real-time photos due to various delays, MDD has a longer delay, which is significant in disaster management. MDD allows for non – static recalculation of routes based on TSP calculations based on the Euclidean distance formula. However, using the Euclidean formula for a catastrophic situation has significant drawbacks. When combined with the sea height and slope of the place, the Euclidean distance formula may yield the most suitable distance between two coordinates. The concept of identifying a water flow route is similar to finding a route for survivors or rescuers in MDD. The researchers created a novel metric called slope from the water point to the nearby reservoirs that is adequate for water flow. To determine the starting solution, this slope will be combined with the distance. As a slope from one place to another, the utility of slope has been evaluated from negative to positive. Three variables are used as inputs to GA for a solution: coordinates of two points, distance, and slope. GA examines fitness to discover the best route, and then delivers a solution following crossover and mutation. Each iteration attempts to identify the minimal fitness after being applied to an individual, which is frequently referred to as a gene in genetic algorithms. In the case of flood management by moving water to the closest reservoir, the goal is to determine the shortest route,

which implies that all routes and slopes are unique. In that case, solutions will be chosen based on their ability to produce offspring, or new solutions.
On the basis if this paper it is fair to say, that after improving some features of the algorithm of MDD, it can be safe to be used in disaster management systems for flood avoidance purposes.

The third approach regarding the problem was offered by Mazin Abed Mohammed, Mohd Khanapi Abd Ghani, Raed Ibraheem Hamed, Salama A. Mostafa, Dheyaa Ahmed Ibrahim, Humam Khaled Jameel, Ahmed Hamed Alallah [17]. The vehicle routing problem (VRP) is one of several complex challenges for which no perfect solution exists. Many researchers have conducted countless studies over the previous few decades, employing a variety of methods and strategies. However, obtaining the lowest cost in any research is quite difficult. They have, however, developed approximation solutions which vary in efficiency based on the search space. The difficulty in this research is the following: there are a lot of trucks that are utilized to transfer applications to a specific location. Every day, each van departs from a central place at a separate time. The truck collects applications from initial coordinates and transports them to the instance site via a variety of routes, returning to the initial location at precise times each day, beginning early in the morning and ending at the conclusion of official working hours, under the following conditions: Each route will visit each place once, and each vehicle's capacity is sufficient for all applications covered in each route. Using the K-Nearest Neighbor Algorithm, this study attempts to determine an ideal route outcome for VRP (KNNA). To acquire an ideal VRP resolution with the objectives as follows: to decrease the distance and time for all routes, resulting in faster client transportation to their destinations; to use the capacitated vehicle routing problem (CVRP) model to optimize the solutions. The method was provided in two stages: first, the algorithms were changed to address the research

topic, which had a different procedure than the standard algorithm. The method's structure is such that it does not call a need for a huge database to keep the population, that speeds up the program's execution to achieve the solution; second, the algorithm has demonstrated its ability to solve the issue and identify the quickest route. The findings of this research revealed the following: A dynamic KNNACVRP universal list; KNNACVRP's assessment measure was identified and developed. The CVRP model is used to optimize VRP services in this study. The K-Nearest Neighbor Algorithm (KNNA) is used to address this issue since it is capable of tackling a wide range of real-world problems. All of the KNNA procedures are performed by the algorithms. The fitness value is computed in a straightforward fashion as a distance's function because the smallest cost of the study issue is dependent on lowering the distance. The KNNA is structured in such a form that the search process is sped up. Despite the problem limits, the algorithm meets the purpose of the study by enhancing the distance of transportation paths. The method was put in application to the issue online to evaluate its validity and reliability, and it was successful in resolving the issue and providing the quickest path in very little amount of time. The availability of a sole variable in the study topic (distance), as well as the small and finite number of station stops, makes finding a solution relatively simple, and does not highlight the KNNA's strength in dealing with complicated and confusing situations. There is a single physical route that connects the various locations for gathering together kids, preventing the option of identifying additional physical routes and weighing them up to find the lowest distance path. Another variable, such as a heuristic function to be regarded as the road's slope or any other factors such as traffic jams, road smoothness, etc, is recommended. The slope will be compounded by the distance between two subsequent bus stops to create a weight. This will assist in determining the precise route time. The goal of the study was to find the optimum solution to the VRP

problem. The purpose was to lower the cost of transportation, which is a service that is supplied to the problem for free. In conclusion, it is proved that the KNNA is effective in solving the VRP and producing estimated results because it is eligible for a wide range of difficulties. The KNNA's strength stems from its ability to be altered to solve any issue by combining multiple ways or adjusting its methods as indicated by the situation, as this study has done.

Another approach I have reviewed was proposed by Gloria Cerasela Crişan, Camelia-M. Pintea and Vasile Palade [8]. The successful management of emergency circumstances requires the strategic design of logistic networks such as highways, trains, and mobile phone networks. Geographic coordinate systems could be utilized to create new traveling salesman problem (TSP) cases incorporating GIS elements. In this paper the researcher writes about a framework for creating a systematic succession of instances. The current research presents a recurring framework for creating a systematic succession of instances using the Lin–Kernighan heuristic. The system aims to simulate real-life random unpropitious events that affect vast areas, such as heavy rains or the entrance of a polar front, as well as focused relief provision in the early stages of a reaction. They utilization of the first Romanian TSP instance containing the main human settlements as a proof of concept for this framework, and generate numerous sequences of instances from it. The goal of this study is to create new approaches for assessing hazards in large-scale, complicated networks. The two primary benefactions of this research (the Romanian GIS-TSP instance and the ALTER-FTSP structure) work together to achieve this goal: the former provides real-world support for the latter. The instance is a complex network of spatial points, and the framework is a flexible and broad description of the network evolution, as the real world operates in Space+Time dimensions. As a result, they are complementary. A situation is described that includes uncertainty. This technique might be employed

in transportation safety: after an earthquake, a swarm of drones would investigate the randomly shattered sensors from a wide territorial road network.

There’s another study by Siba Prasada Tripathy, Samarjit Kar and Tandra Pa, who suggested a CSP algorithm. The rescue or relief team tries to service each of the areas in the impacted area during a humanitarian relief operation and mass fatality management [24]. It is impossible for the crew to reach every node in a single effort due to a lack of time or resources. As a result, traveling some of the locations and inviting locals to the visited location is a better strategy to complete the assignment. On a given completely connected graph, the Traveling Salesman Problem (TSP) aims to determine the least cost Hamiltonian path. There are multiple TSP versions that consider various aspects and are solved using various approaches. TSP is a generalization of the Covering Salesman Problem. J. R. Current and D. A. Schilling first developed CSP in Transportation Science, vol. 23, where the goal is to discover a Hamiltonian tour with a minimum length that visits a subset of nodes while maximizing the covering nodes residing within a predetermined distance but not in the tour [13]. The authors created a modified Metameric Genetic Algorithm (MGA) [22] for CSP that includes a new crossover operator called the Global Parent Crossover operator (GPX).

The suggested MGA is then applied to 16 normal TSP instances using the GPX operator. Each customer covers its 7, 9, and 11 closest customers in the usual instances, resulting in 48 instances. The new MGA's findings were then compared to two existing methods: Current and Schilling, as well as Memetic Algorithm. The results show that the suggested MGA GPX heuristic outperforms the Current and Schilling heuristics for covering salesman problems with all customers covered. The suggested technique outperforms the Memetic algorithm by 25% in terms of execution time, but it increases the length of the tour by 14%. The suggested CSP algorithm's purpose is to cover all consumers in a devastated area by traversing a

subset of facilities, where a node can be either a customer or a facility, and it can be used in real-world settings such as after natural or man-made disasters. The results demonstrate that the suggested metaheuristic algorithm outperforms two other algorithms. This challenge can be extended in the future for uncertain environments where the cost of the edges or the tour demand are not represented by clear quantities.

Among the papers I reviewed, in B Fernandez's study of "Travelling salesman problem: Greedy single point crossover in ordinal representation", the author compares GSPC, TPC and SPC [11]. Genetic Algorithms (GA), Simulated Annealing (SA), and Tabu Search (TS) are the most well-known metaheuristic algorithms [1]. The choice of genetic operators, particularly selection, crossover, and mutation, has an impact on GA performance [2]. In GA, the crossover operator is crucial since it is utilized to transmit information during the solution search [5]. One of the most basic crossover operators is single point crossover (SPC). The path form is the most used representation for solving TSP problems with Genetic Algorithms, due to its intuitive representation and positive performances. Unfortunately, this representation cannot be used with traditional crossover operators since the ensuing children may have redundant alleles, resulting in the loss of another point, which is incompatible with the TSP notion. Ordinal representation coding can be used with traditional crossover and mutation operators; however the experimental results are mixed. A greedy algorithm is one that, when addressing a problem, always chooses the option that appears to be the best at the time; the option is optimal locally in the hopes of leading to a globally optimal solution [21]. The efficiency of GA in the ordinal representation coding scheme has to be enhanced when applying the greedy algorithm on the SPC operator to obtain the global optimal solution. The results of this study's tests demonstrate that the GSPC operator has the best fitness, however it takes longer to

compute. TPC provides the quickest computation time. Because the testing of candidate two alleles from parents is done before allele exchange, GSPC prevails in best fitness. This results in a long computation time. TPC has a short computation time because the allele exchange between the two parents occurs from the first to the second point, while the second point is not invariably the site of the last gene, as is the case with SPC.

In final paper I reviewed, by M. Pallin, J. Rashid and P. Ögren, covering "A Decentralized Asynchronous Collaborative Genetic Algorithm for Heterogeneous Multi-agent Search and Rescue Problems," [20], the researchers propose a highly decentralized version of the Genetic Algorithm (GA) for combined task assignment and path planning, in which each agent only knows its own capabilities and data, as well as a set of so-called handover values communicated to it from other agents over an unreliable low bandwidth communication channel. These handover values are used with a local GA with no other agents to choose which tasks to perform and which to delegate to others. They compare our technique to a centralized version of GA and a partially decentralized version of GA in which calculations are local but all agents require comprehensive information about all other agents, such as position, range, battery, and local obstacle maps. We analyze the three algorithms' solution performance as well as the messages transmitted, and find that the suggested algorithms have a slight performance loss but a large reduction in necessary communication.

As one of the main goals of the emergency rescue team is fast response, in this study I decided to implement genetic algorithm in order to achieve the optimal task arrangement between the teams. To try to decrease the calculation time, instead of the randomly choosing the parent chromosomes for pairing, KNN algorithm will be

used. This will provide the crossover with more diverse chromosome pairs, which will lead to creation of more diverse offsprings.

3. Implementation

3.1. Methodology

The code was implemented on Jupyter Notebook. The chosen programming language was Python version 3. At first, to simulate the emergency situation we need to create an environment for the said situation. Suppose there are 2 emergency centres in the town. The main office that receives calls for the mentioned centres receives 5 calls in total. For simplicity, let's assume that the centres are equipped equally and the knowledge, experience and the ability of the rescuers are the same. To assign the tasks between these teams and the locations that the rescuers will go first, the Hybrid Multi-Objective Genetic Algorithm was implemented. The risks that will be considered in the algorithm are calculated as described in the situations below. There are 10 locations the rescuers teams should go to and the situations include:

1. 1 person. 60 years old, male. Has a heart attack. Had previous issues with regard to his heart condition. Is still conscious. The life threatening risk from heart attack is very high, but considering the patient is still conscious, the risk rate will be considered as 8 for this case.
2. 1 person. 34 years old, male. Has deep cut in his arm leg. The injury was received at a construction site. This increases the level of infection, which may spread out from the open wound, which increases the risk rate. The person is half-conscious and lost 400ml of blood. The cut didn't injure the artery, but the vein was damaged severely. Considering the conditions mentioned previously, the risk rate will be put up as 7.

3. 2 people. First one is 52 years old woman. No previous health related conditions. Symptoms of stroke were described. The woman is already unconscious for 5 minutes. The average time the help should be provided to the patient is 3 hours. For some cases, the patient can be saved even after 4.5 hours. This time is considered for only saving the life of the patient. For the patient to have as little consequences from the stroke as possible, the help should be provided within 1 hour. Since the risk for the life of the person with a stroke is already very high, and for the reason that the patient is already unconscious, which could mean that the person might have the medical condition for approximately somewhere between 20 minutes to 40 minutes, which leads us to the fact that the help should be provided in range between 20 minutes to 40 minutes. This lead us to the result that the risk for this person's life is as high as 9.

   The other person at the same location is the first patient's daughter, 20 years old, who got accidentally cut while trying to catch the first patient from falling. The cut didn't damage the artery but was very deep regardless. The patient lost a relatively dangerous amount of blood. Considering the situation, the risk for this patient is calculated to be 7.
4. In the fourth location, the patient is a 7 years old child, who accidentally swallowed a cap of a pen and is having difficulties breathing. Because the caps have holes on the tops of them to prevent suffocation, the patient can still breathe, but regardless, the cap of the pen blocks a big portion of the airway. On top of that, the panic that comes with the inability to take a deep breath, which leads the adrenaline in the child's blood to rise, causing higher heart beats per minute, which as a result causes quick and short breaths. Due

to the situation being not seriously damaging and life threatening, the risk of the life of the person will be average, and in number it will be 6.

5. The fifth location has 3 people needing immediate help. A group of friends just returned from a vacation from a country that had some type of infection. The ages of these people are 25, 26, and 26, and are male, female, and male respectively. The youngest of them all has a seizure, and the other 2 have high fever of 38.7 and 39.5 C respectively. The person with a seizure, has the grand mal seizure, with is one of the type that can lead to unconsciousness and causes violent muscle contractions. The patient is still conscious, but has had seizure for 10 minutes already. Typically, seizures will cause damage to the brain after 30 minutes from the start of the seizure, which means that to prevent it, the help to the patient should be provided in not more than 20 minutes.

   On the other hand, there are 2 more people that need help. The body temperature of these people is high but because it is not higher than 40^{o}, there is no life threatening risk to the moment, their bodies are just trying to get rid of the infection they caught, so the risk will be considered 3.
6. The next location has 2 people, one of them is 29 years old, the other one is 39 years old. Both of these people have broken bone. The wound is closed, no blood loss, and both are conscious. The risk for this case is low, and for both will be considered as 2.
7. This site has only 1 person who has mild headache. Because there are no other symptoms, the risk for this case will be 1.
8. The 8^{th} location also has 1 person. The patient who suffers from allergies has gotten an anaphylactic shock. The symptoms include suffocation, purple skin. The patient is still conscious, but can barely breathe. This means that the risk for this case will be 8.

9. In the 9th location, a teenager got a high fever of 41°C. This temperature is considered life threatening, and needs immediate intervention, so this case will have the risk of 8.
10. The last location has 1 person and she is having a seizure. She has been unconscious for 2 minutes by the time of the call. This leads to the result of risk being 9.

3.1.1. Data storage

To store and later use the input values the data will be stored in DataFrames. There will be 6 tables in total:

1. Table with all of the solutions that are being processed. The tables will include columns 1st center, 2nd center, generation, fitness value, Solution number and Class

   1st center will contain the list of location in order of the help provided by the first rescue team.

   2nd center will contain the list of locations in order of help provided by the second rescue team.

   Generation – is the age of the chromosome. This information is needed in the selection phase where the KNN will be implemented.

   Fitness value – the fitness value will be calculated by Equation 2.:

$$\text{Min} \sum_{i=1}^{n} w_0 t_{ij} + w_1 r_i + \frac{w_2}{n_i} + w_3 v_i$$

Eq. 2, Fitness Function.

   where w is the weight of each variable, t_{ij} is the time it takes to move from the rescue center to the accident location or from one accident location to another one, r_i is the level of risk of people's lives, n_i is the number of people in the location i and v_i is the amount of time that will be spent during the operation.

The weights in the function are given in Table 1.:

Table 1. The weighs used in the

Weight	Value
w1	0.7
w2	0.82
w3	0.78

w1 shows the importance of time it took to reach to the location from the center or to the other location. The time it took to reach the patient will affect the future condition of the person a lot.

w2 shows the importance of the risk that threatens the patient's life. The time spent on the road is, of course, very crucial, but if the patient with headache located in 10 minutes ride and a patient with heart attack is 30 minutes away, the rescue team should first head to the patient with the heart attack.

w3 shows the importance of the number of people at the site. It is obvious, that the number of people, the help will be provided to and who will be saved is very important.

2. Another DataFrame table will contain the information about all of the possible injuries and situations. The columns will consist of ID, Name, Risk, and Time.
ID – the id of the injury or situation. This will be used to link the information about the injuries with other tables during the calculations.

   Name – the name of the injury or the situation. The name can be used to identify the injury by the user.

   Risk – all of the injuries will have risks defined beforehand. This column will be used to calculate the risk for each of the solutions afterwards. The more the risk, the higher the number will be.

   Time – this variable shows the time needed to spend to treat each of the injuries or situations. This number will also be included in the Risk calculation of the solution.

3. A separate table contains the location of the situations that need help. The table columns will contain id, Number, Injuries
   Id – the id number of the location. This will be used to link the information about the situations with other tables during calculations.

   Number – number column will show the number of the people at site.

   Injuries – The ID's of the injuries in this table will be presented in this column. Due to the fact that some of the situations may have more than one person who needs immediate help, all of the injuries will be included in a form of a list.

4. One more table will contain the information about the time needed to get from one location to another one. There will be 10 locations in total, so there will also be 10 columns and 10 rows.
5. Another table will contain the information about the time needed to get from the centres to the emergency situation locations. Since there will be 10 locations and 2 centres, the number of rows will be 2 and the number of columns will be 10.
6. The last DataFrame table will consist of the variables needed to be used to calculate the fitness value. The columns are solution, Time_center, Time_location, Risk and Number.
   Solution – solution will show the id of the solution, the variables of which are being calculated.
   Time_center – The time it will take to get from the centres to the locations for the defined solutions.
   Time_location – The time needed to get to the location from the previous location. This variable will include time between the locations and the time needed for treatment of the previous patient or patients.
   Risk – The variable identifies the total amount of risk for each of the situations. As the time passes, the risk of each of the injured person's life

increases. For this reason, each next location will have the risk added the multiplied by 1.3.

Number – the number of people in the site that need help.

First, 100 random solutions were generated. Then their fitness values were calculated. Now the Selection phase of the genetic algorithm should begin. There are two classes the solutions will be divided into. These classes will be used to select the random solutions for crossover. To train the model, the train data was constructed. The train model consists of 4 columns. The first two columns are used to show the generation, which is the age of the solution, and the fitness values of the solutions. In the code, they will be considered as the X_train part. X_train part are the variables that determine the label of the class. 100 rows of train data were provided to initialize the training data's target. The Target column will contain the class of the solution, based on the X_train data. The Target column will be the y_train in this case. Based on this training data, and the fitness function calculated before, the solutions, which were randomly generated before, will be assigned to a class.

Solutions in the training data will be used to link the table used for KNN and the solutions table. To find the class of the solutions, the generation, fitness value, class and the solution id will be added to the train data table. The code will divide the test data and the train data based on the Class column. By default, the class of the solutions created is NaN, which is the equivalent of null in numpy library of python. The algorithm will check the values for the solutions in the column, and if the value of the class of the solution is NaN, the solution will be treated as test data, which needs to be classified, if there is some value, the solution will be treated as train data, and the class will not be changed. To merge the table with the solutions table that will have the main information needed to preform Genetic Algorithm

techniques, the Solution of the data in the train data will be considered by default as -1, so these rows will not be transferred to the solutions table.

The number of the nearest neighbours in the algorithm will be taken as 3, so the class of the solution will be decided based on 3 of the points in train data.

Now that we have our solutions classified, the data will be divided into 2 parts, 1 part will contain solutions of class 1, and the other will contain the solutions of class 0. This is the most important part of the algorithm that makes it different from previously proposed ones. It is biologically proven that close relatives' children can inherit more diseases that are common to the parents, and also, they can inherit their negative genetic traits. This means, that statistically, the probability of children with more diverse background is higher that of those, who are related. This statement was the inspiration of the proposed algorithm. After the solutions were divided into 2 classes, the algorithm takes random solutions from different classes and performs crossover operation. Because the locations are stored in 2 different lists, which helped with identification of the boundary between the lists that are related to each of the centres, I combined them into 1 list. So, for example, if the lists were [1, 5, 2, 6, 8] and [0, 3, 7, 9, 4] for the first centre and [4, 2, 5, 1, 0] and [6, 7, 3, 9, 8] for the second one, to perform crossover, the lists will be presented as [1, 5, 2, 6, 8, 0, 3, 7, 9, 4] as the first parent and [4, 2, 5, 1, 0, 6, 7, 3, 9, 8] as the second parent. The crossover method used in the proposed algorithm is going to be one point crossover. In this case, since I combine the lists of the rescue centres, the crossover point is also going to be the boundary for determining the split border for the lists of the rescue teams' arrangement. So, as an example, for the parent chromosomes mentioned above, the crossover operation will be performed in the following way:

[1, 5, 2, 6, 9, 0, 3, 7, 8, 4]

[6, 2, 5, 1, 0, 4, 7, 3, 9, 8]

Randomly generated crossover point will be 5

This means that the newly generated chromosomes will be

[1, 5, 2, 6, 9, 4, 7, 3, 9, 8]

and

[6, 2, 5, 1, 0, 0, 3, 7, 8, 4]

And the division between the rescue teams will be [1, 5, 2, 6, 9] for the first teams and [4, 7, 3, 9, 8] for the second team for the first solution and [6, 2, 5, 1, 0] for the first team and [0, 3, 7, 8, 4] for the second rescue team. As we can see in this example, the results don't include all of the locations. This part could be changed during the mutation algorithm. The mutation operation as described in the introduction part helps to add some diversity in the results. In the case describes above, the mutation operation can also help eliminate the problem of the repeating locations in the list. The mutation method in the proposed algorithm will take a random location from the solution and replace it with a random number in the range of the locations, which in this case is a digit between 0 and 9. When applied to the example mentioned above, the result of the mutation can change the solution from a non-valid one to a valid one:

The first solution is [1, 5, 2, 6, 9], [4, 7, 3, 9, 8]. As we can see, the location number 9 repeats here twice. Let's suppose that the mutation randomly took the 4^{th} location of the 2^{nd} centre and changed it to 0. In this case, the solution after going through mutation problem will give us the result of [1, 5, 2, 6, 9], [4, 7, 3, 0, 8]. As we can see, the operation turned the solution that couldn't be used in real situation, to a valid one, where all of the locations will be visited.

The second solution is also not a valid solution yet. In the second solution [6, 2, 5, 1, 0], [0, 3, 7, 8, 4] we also see that the location number 0 is repeating itself. The mutation operation is again performed in to the solution. The algorithm takes a random location, which let's assume is the 3^{rd} location of the 1^{st} rescue

teams' tasks, and picks a random value from 0 to 9 that will be 8. When applied to the solution, we get the result [6, 2, 8, 1, 0], [0, 3, 7, 8, 4]. In this case, the solution is still not valid, because not only after the mutation phase we still have repeating 0's, but now we also have repeating 8. This, and the fact that the locations number 5 and 9 are not assigned to any of the rescue teams, means that the solution is not valid and therefore, cannot be included to the population. To check whether the solution is valid or not, the valid_solution function was created. The solution consisting of 2 lists, is combined into 1 list first. Then, a counter is implemented for each of the location numbers. If any of the locations have a count more than 1, the function will return False as a result. After the crossover operation and the mutation operation will be implemented to the available solutions, the valid_solution function will be used.

The Genetic Algorithm method used for the proposed algorithm is Steady State Genetic Algorithm, or simply SSGA. As mentioned in the beginning of the thesis paper, in this type of Genetic Algorithm, two parent chromosomes will be taken first. Then after the crossover and mutation operations will be applies, the solutions will be added to the initial population by replacing the ones with the least fitness function. In addition to the genetic algorithms operators mentioned, the valid_solution function will also be used. It will be used when the algorithm will try to replace the solution with the least fitness function, to the newly generated ones, but first it will check if the solution is valid for the problem. It is possible that one of the solutions will be valid, but the other will not, as shown in the example above. In this case, the first solution, that is valid, will be included to the new population, but the second one, which doesn't meet the criteria, will not be included.

3.2. **Simulation and experimental results**

In the 10 locations described above, the initial solutions will be randomly generated to be digits from 0 to 9. The locations will be distributed between 2 centers in a form of a list. By default, the initial solutions will have the fitness value of 0 and the classes will be NaN from numpy.

The scenario described above, with the locations ids, the risks for the life of the patients will be manually included in the DataFrame in a form of a multi-dimensional list like:

```
Loc_data = [[0, 1, [1]],
            [1, 1, [3]]
            …
            ]
```

where the each of the lists in the list will describe 1 of the location situation. For example, 0 will be the id of the location, 1 is the number of people that have injuries or medical conditions and the last list is going to be the injury ID's.

The injury ID's are required to find the risk of the injuries and conditions and the time that needs to be spent for saving the patient, or provide help.

The risks for the people's lives will be taken from the prepared table. This table will be the same for all of the scenarios, and contain all of the possible injuries and medical conditions. The example of the table with all of the injuries, their names, ID's, and the time needed is shown in Table 2.:

Table 2. Example of the injuries and situation tables

id	Name	Risk	Time for treatment
1	Heart Attack	8	40
2	Stroke	9	30
3	Deep cut	7	60
4	Suffocation	6	20
5	Fever	3	10

The distance between the locations and the centres will be measured in time required to get from the start point to the destination point. For the simulation, the values will be inserted manually, but in the future, when the algorithm will be used, the times can be calculated from the maps predictions, for example Google Maps. The times between locations for this example will be as shown in Table 3:

Table 3. Example of the locations distance table

Locations	0	1	2	3	4	5	6	7	8	9
0	0	2	2	4	3	4	2	3	5	1
1	1	0	4	5	1	5	2	5	2	4
2	2	3	0	4	2	3	2	3	4	1
3	5	4	4	0	2	4	2	5	4	4
4	3	1	2	2	0	3	5	2	1	2
5	4	1	2	4	3	0	2	3	5	1
6	1	2	4	5	1	5	0	5	2	4
7	2	3	1	4	2	3	2	0	4	1
8	5	4	2	3	2	4	2	5	0	4
9	3	5	2	2	6	3	5	2	1	0

The next step is defining the table, which will contain information about the time required to travel from the rescue centres to the locations of the situations. An example for the scenario mention above will be shown in Table 4:

Table 4. Example of the centres to locations distance table

location / centre	0	1	2	3	4	5	6	7	8	9
1st	2	2	4	1	3	4	2	4	1	2
2nd	4	3	1	6	2	3	4	5	1	4

3.2.1. Scenario 1

After filling all of the tables, the algorithm can now be run according to the specific scenario. The example below is related to the scenario described above.

The total number of runs done for of the scenarios will be 10. The number of iterations of the Genetic Algorithms main loop for each of the runs will be 5000. The weights for all of the scenarios are mentioned in Table 1. For these 10 runs, the lowest number of generations needed to get the highest fitness value was 927, the maximum was 1219. The average for the 10 runs was 1034. The comparison can be seen in Figure 4.

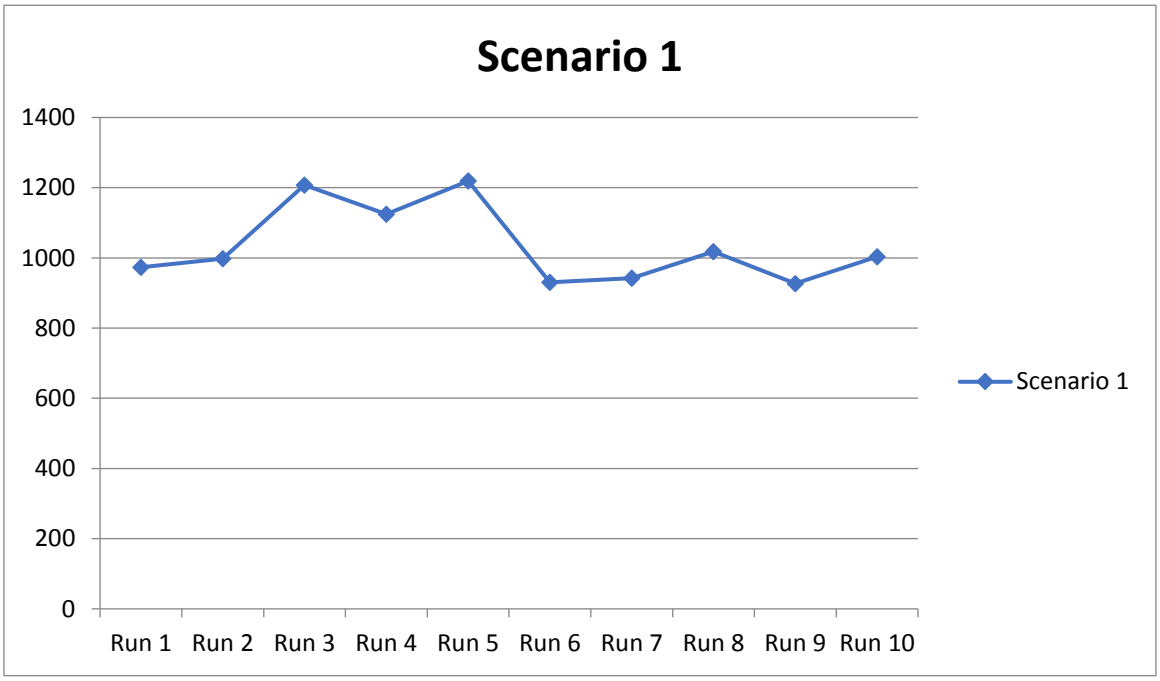


Figure 4. Generations with the best fitness values in Scenario 1 for each execution

The best solution for this scenario was the chromosome [[2, 0, 7, 4, 1], [9, 3, 4, 8, 6]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 29.743926 and was reached in the 6th run of the code. The generation needed to reach the value was 927.

3.2.2. Scenario 2

For the second scenario, the following situations have been taken into consideration:

1. The 1st location has 1 person. The injury sustained is broken arm. The wound is close; there is no blood loss, so the risk for this case is evaluated as 2.
2. The 2nd location has also 1 person. The patient is suffering from an allergic reaction. The skin is starting to turn purple; the patient has serious difficulties breathing, but is still conscious. The risk for the life in this case will be evaluated as 8.
3. The 3rd case is the location with 1 person having a heart attack. The person is still conscious but had several heart related conditions in the past. The risk for this case will be considered as 8.
4. The 4th location will have a person with a relatively strong headache. No other symptoms were detected, so the risk for this case will be evaluated as 2.
5. The 5th location has 2 people. They have been in a fight, 1 has broken rib, and the other one has a deep cut. For the first injury, the damage from the broken rib part was not life-threatening so the risk will be evaluated as 2. The second patient with the deep knife was attacked with a knife. The cut didn't touch any of the vital organs, so the risk of this case will be evaluated as 7.
6. The 6th location has 1 person, an elderly person, who is going through a stroke. The person is conscious but all of the symptoms are highly defined. The risk of the injury will be evaluated as 8.
7. In the 7th location, the person is having a stomach ache. The suspicions are that the person has eaten rotten food. Since the only symptom is the pain, the risk will be evaluated as 2.

8. The records from the 8th location show that there is a child, who is having a seizure. The patient has been unconscious for 6 minutes. In this case, the risk will be evaluated as high as 9.
9. The 9th location has a person with high fever. The value already reached 40.8° C. Since this temperature is considered as dangerous, the risk for the person's life will be as high as 8.
10. The last location has 2 people with fever. None of them reached 40° C, so the risk will be evaluated as 3.

The scenario was again run 10 times. The results of these executions show that the fastest the algorithm reached its highest fitness value is 873, the slowest is 1025 and the average number is 938.

The comparison can be seen in Figure 5.

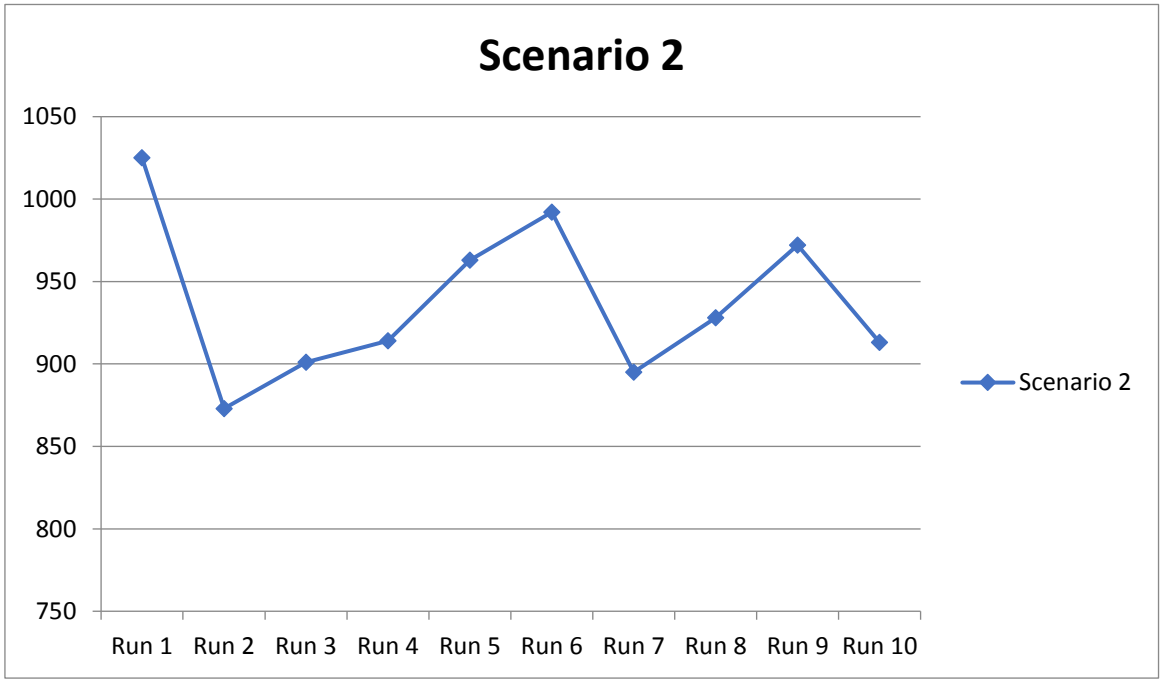


Figure 5. Generations with the best fitness values in Scenario 2 for each execution

The best solution for this scenario was the chromosome [[7, 8, 9, 3, 0, 6], [5, 4, 1, 2]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 31. 426823 and was reached in the 2nd run of the code. The generation needed to reach the value was 873.

3.2.3. Scenario 3

The scenario number 3 has the following situations:

1. The person is having a heart attack. Unconscious. No previous heart related conditions. The risk for this person's life will be evaluated as 8.
2. The second person is also having a heart attack. She is conscious; no heart related conditions were found before. The age is 23. The risk for this person life will be evaluated as 7.5.
3. The third location has 4 people. 2 of them have deep cuts, and the other 2 have 2nd degree burns. The risk for the life of the people with the cuts is 7 for both, since the cuts didn't touch any vital organs. The risk for both of the people with burns is evaluated as 5.
4. The 4th location has 1 person. He is going through an allergic reaction. The patient is still conscious, had difficulties breathing, but the skin is in a normal colour. The risk for this person's life will be evaluated as 7.
5. The 5th location is at the beach. 2 people are drowning, 1 person tries to save the other one. Both of the people's lungs are filled with water. The risk of both of these people's lives is rated as high as 9.
6. The 6th location has a person with stroke. The patient is unconscious, and has been in that state for 30 minutes. The risk for the life of this patient is 10.
7. The 7th location has 2 people. Both of them have high fever, more than 40.5^{o} C. The risk for both of this people will be evaluated as 8.
8. The 8th location has 1 person. The patient is having a mild seizure. The symptoms are not life threatening so the risk for the person's life will be 5.
9. The 9th location is a bank. There has been a mass shooting, and there are 3 injured people. The injuries include gunshot wounds and knife wounds. The patients lost a lot of blood. The risk for this case will be 9

10. The last location has a person with a failed suicide attempt. The person tried to take a dangerous amount of tablets. The patient is unconscious. The heart rate is below 40 beats per minute. The risk for person's life is 10.

The scenario was again run 10 times. The results of these executions show that the fastest the algorithm reached its highest fitness value is 923, the slowest is 1113 and the average number is 998. The comparison can be seen in Figure 6.

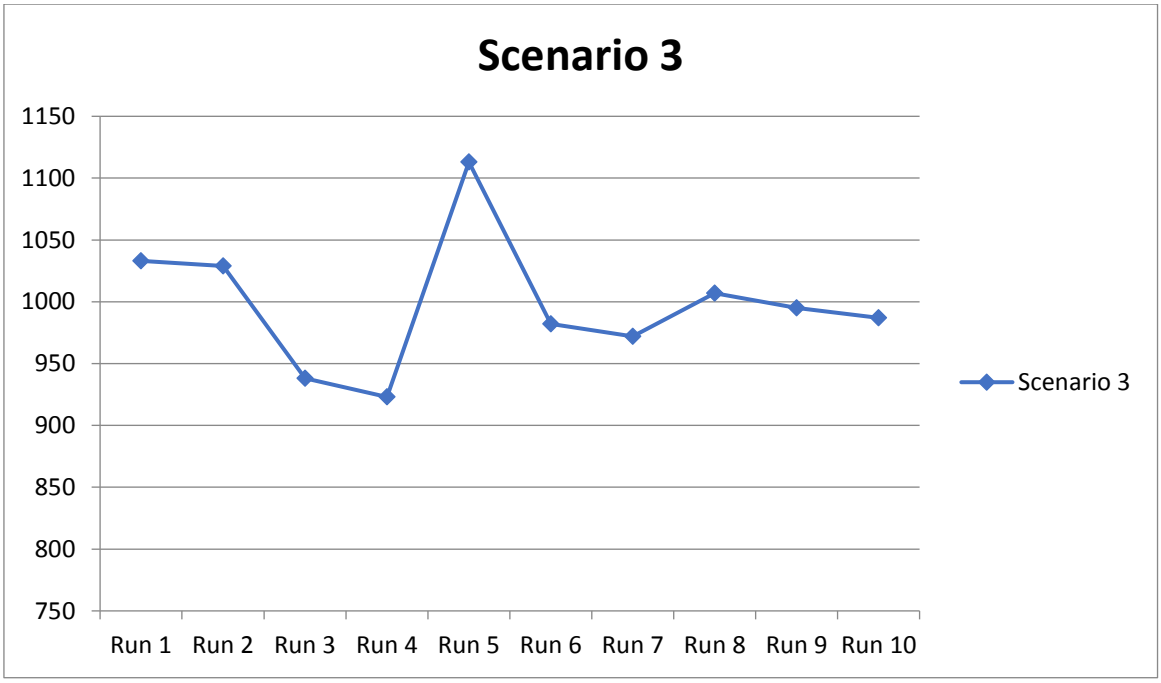


Figure 6. Generations with the best fitness values in Scenario 3 for each execution

The best solution for this scenario was the chromosome [[5, 4, 2, 1, 0], [8, 9, 6, 3, 7]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 45.625383and was reached in the 4th run of the code. The generation needed to reach the value was 923.

3.2.4. Scenario 4

The scenario number 4 has the following situations:

1. The 1st site has 1 person. The emergency situation is a heart attack. The patient is an elderly person. The risk for the patient's life is considered 8.

2. The 2nd location also has 1 person. The injury is broken leg. The wound is open, so the risk increases and will be evaluated as 6.
3. The 3rd location has a person with a stroke. He has been unconscious for 2 minutes. The risk for the life of the patient is as high as 9.5.
4. The 4th location has a teenager with a seizure. The patient is conscious and has mild convulsions. The risk will be evaluated as 6.
5. The 5th location is a gas station. The explosion left 2nd degree burns on 2 people. The risk for both people will be 5.
6. The 6th location has a patient with dislocated shoulder. The pain is medium. The risk will be taken as 2.
7. The 7th location has a person with a heart attack. The person is young, and conscious. The risk will be 7.5.
8. The 8th location is the base of a mountain. An alpinist fell while climbing. Several bones have been broken. The risk is considered to be 8.
9. The 9th location has 1 patient, which is going through a stroke. The patient is an elderly person, so the risk increases to 9.
10. The last location has a person with a deep cut. The cut didn't touch any of the vital organs, but there is a lot of blood lost. The risk will be evaluated as 7.

The scenario was again run 10 times. The results of these executions show that the fastest the algorithm reached its highest fitness value is 902, the slowest is 1157 and the average number is 1004. The comparison can be seen in Figure 7.

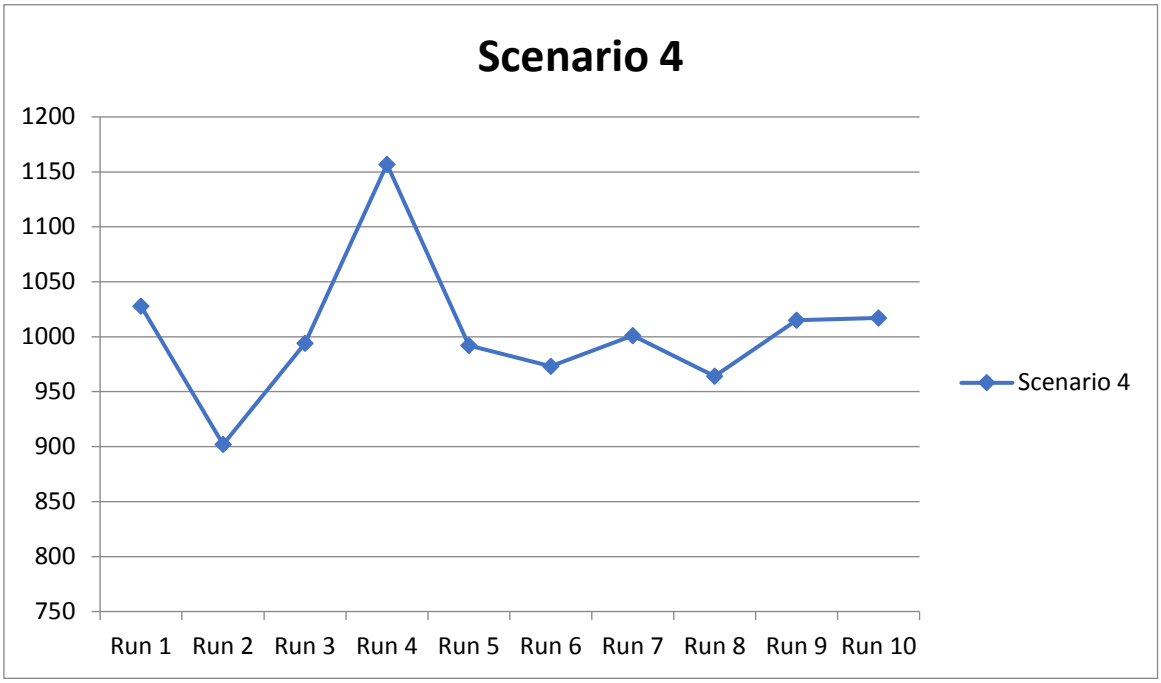


Figure 7. Generations with the best fitness values in Scenario 4 for each execution

The best solution for this scenario was the chromosome [[2, 6, 0, 3, 1, 5], [8, 9, 4, 7]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 37.945392 and was reached in the 2nd run of the code. The generation needed to reach the value was 902.

3.2.5. Scenario 5

The scenario number 5 has the following situations:

1. The 1st location has a person with a headache, nausea and dizziness. The condition has been continuing for 2 hours and is getting worse. The risk for this case will be evaluated as 5.
2. The 2nd location has a person with a heart attack. The person is young, and conscious. The risk will be 7.5.
3. The third location has 4 people. 2 of them have deep cuts, and the other 2 have 2nd degree burns. The risk for the life of the people with the cuts is 7 for both, since the cuts didn't touch any vital organs. The risk for both of the people with burns is evaluated as 5.

4. In the 4th location, the person is having a stomach ache. The suspicions are that the person has eaten rotten food. Since the only symptom is the pain, the risk will be evaluated as 2.
5. The 5th location will have a person with a relatively strong headache. No other symptoms were detected, so the risk for this case will be evaluated as 2.
6. The 6th location has 1 person. The injury sustained is broken arm. The wound is close; there is no blood loss, so the risk for this case is evaluated as 2.
7. The 7th location has 1 person. The patient is having a mild seizure. The symptoms are not life threatening so the risk for the person's life will be 5.
8. The 8th location has 1 person, an elderly person, who is going through a stroke. The person is conscious but all of the symptoms are highly defined. The risk of the injury will be evaluated as 8.
9. The 9th location has a person with a deep cut. The cut didn't touch any of the vital organs, but there is a lot of blood lost. The risk will be evaluated as 7.
10. The last location is the main road. A person has been hit by a car. The injury is not serious, no bones were broken. The risk will be evaluated as 3.

The scenario was again run 10 times. The results of these executions show that the fastest the algorithm reached its highest fitness value is 945, the slowest is 1253 and the average number is 1103. The comparison can be seen in Figure 8.

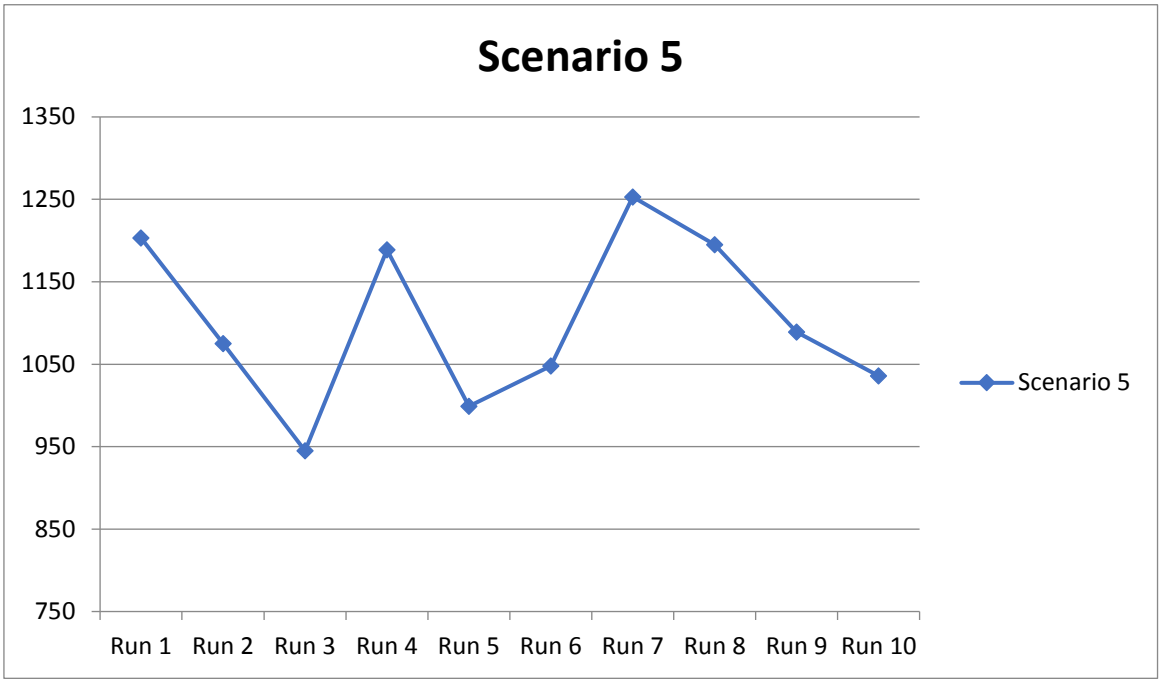


Figure 8. Generations with the best fitness values in Scenario 5 for each execution

The best solution for this scenario was the chromosome [[8, 1, 9, 4, 3, 5], [2, 7, 0, 6]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 48.943021and was reached in the 2nd run of the code. The generation needed to reach the value was 945.

3.2.6. Scenario 6

The scenario number 6 has the following situations:

1. The 1st location will have a person with a relatively strong headache. No other symptoms were detected, so the risk for this case will be evaluated as 2.
2. The 2nd location has 2 people. They have been in a fight, 1 has broken rib, and the other one has a deep cut. For the first injury, the damage from the broken rib part was not life-threatening so the risk will be evaluated as 2.

The second patient with the deep knife was attacked with a knife. The cut didn't touch any of the vital organs, so the risk of this case will be evaluated as 7.

3. The 3rd location has 1 person, an elderly person, who is going through a stroke. The person is conscious but all of the symptoms are highly defined. The risk of the injury will be evaluated as 8.
4. The 4th location has 1 person. The patient is having a mild seizure. The symptoms are not life threatening so the risk for the person's life will be 5.
5. The 5th location is a bank. There has been a mass shooting, and there are 3 injured people. The injuries include gunshot wounds and knife wounds. The patients lost a lot of blood. The risk for this case will be 9
6. The 6th location has a person with a headache, nausea and dizziness. The condition has been continuing for 2 hours and is getting worse. The risk for this case will be evaluated as 5.
7. The 7th location has a person with a heart attack. The person is young, and conscious. The risk will be 7.5.
8. The 8th location is a restaurant. 6 people have been poisoned. The risk for 4 of these people is 2, the risk for the remaining 2 is 4.
9. The 9th location has 1 person. The patient is having a panic attack. The risk is 1, because of the little damage to the patient's health.
10. The last location has a patient with dislocated shoulder. The pain is medium. The risk will be taken as 2.

The scenario was again run 10 times. The results of these executions show that the fastest the algorithm reached its highest fitness value is 934, the slowest is 1044 and the average number is 993. The comparison can be seen in Figure 9.

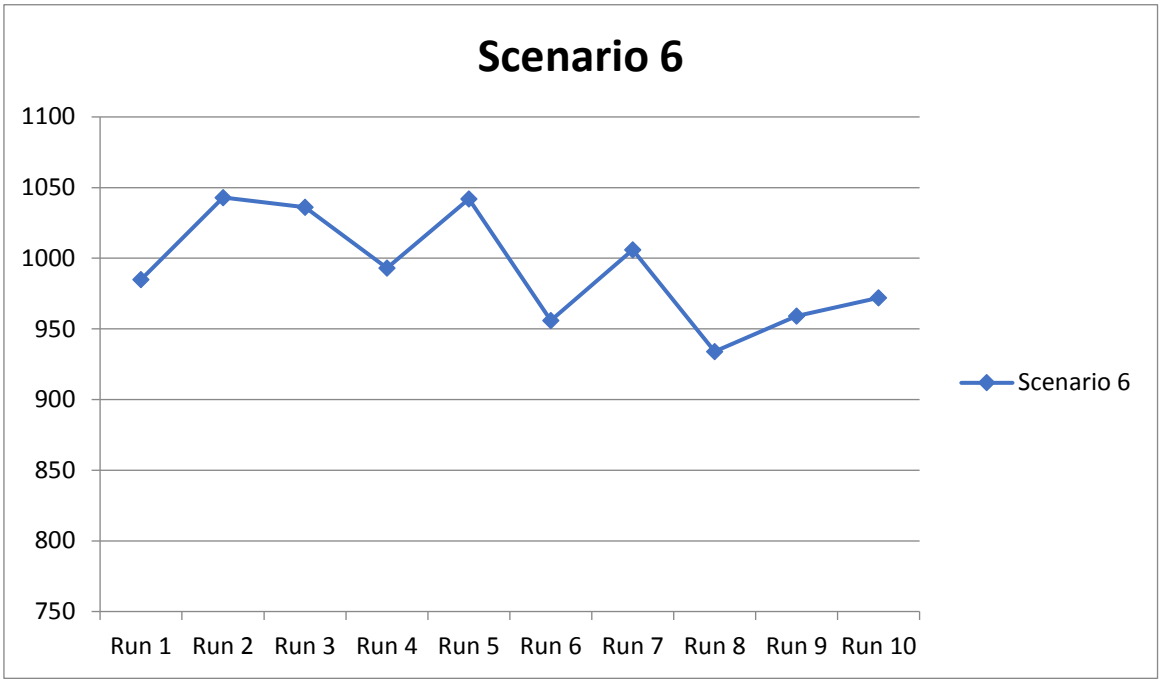


Figure 9. Generations with the best fitness values in Scenario 6 for each execution

The best solution for this scenario was the chromosome [[4, 6, 1, 5], [2, 7, 3, 0, 9, 8]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 37.563781 and was reached in the 8th run of the code. The generation needed to reach the value was 934.

3.2.7. Scenario 7

The scenario number 7 has the following situations:

1. The 1st case is the location with 1 person having a heart attack. The person is still conscious but had several heart related conditions in the past. The risk for this case will be considered as 8.
2. The 2nd location has 1 person. The injury sustained is broken arm. The wound is close; there is no blood loss, so the risk for this case is evaluated as 2.
3. The 3rd location has 4 people. 2 of them have deep cuts, and the other 2 have 2nd degree burns. The risk for the life of the people with the cuts is 7 for both,

since the cuts didn't touch any vital organs. The risk for both of the people with burns is evaluated as 5.

4. The 4th location has 1 person. He is going through an allergic reaction. The patient is still conscious, had difficulties breathing, but the skin is in a normal colour. The risk for this person's life will be evaluated as 7.
5. The 5th site has 1 person. The emergency situation is a heart attack. The patient is an elderly person. The risk for the patient's life is considered 8.
6. The 6th location also has 1 person. The injury is broken leg. The wound is open, so the risk increases and will be evaluated as 6.
7. The 7th location has a person with a deep cut. The cut didn't touch any of the vital organs, but there is a lot of blood lost. The risk will be evaluated as 7.
8. The 8th location has a person with a heart attack. The person is young, and conscious. The risk will be 7.5.
9. The 9th location has 2 people. They have been in a fight, 1 has broken rib, and the other one has a deep cut. For the first injury, the damage from the broken rib part was not life-threatening so the risk will be evaluated as 2. The second patient with the deep knife was attacked with a knife. The cut didn't touch any of the vital organs, so the risk of this case will be evaluated as 7.
10. The 10th location has 1 person. The patient is having a panic attack. The risk is 1, because of the little damage to the patient's health.

The scenario was again run 10 times. The results of these executions show that the fastest the algorithm reached its highest fitness value is 917, the slowest is 1147 and the average number is 1032. The comparison can be seen in Figure 10.

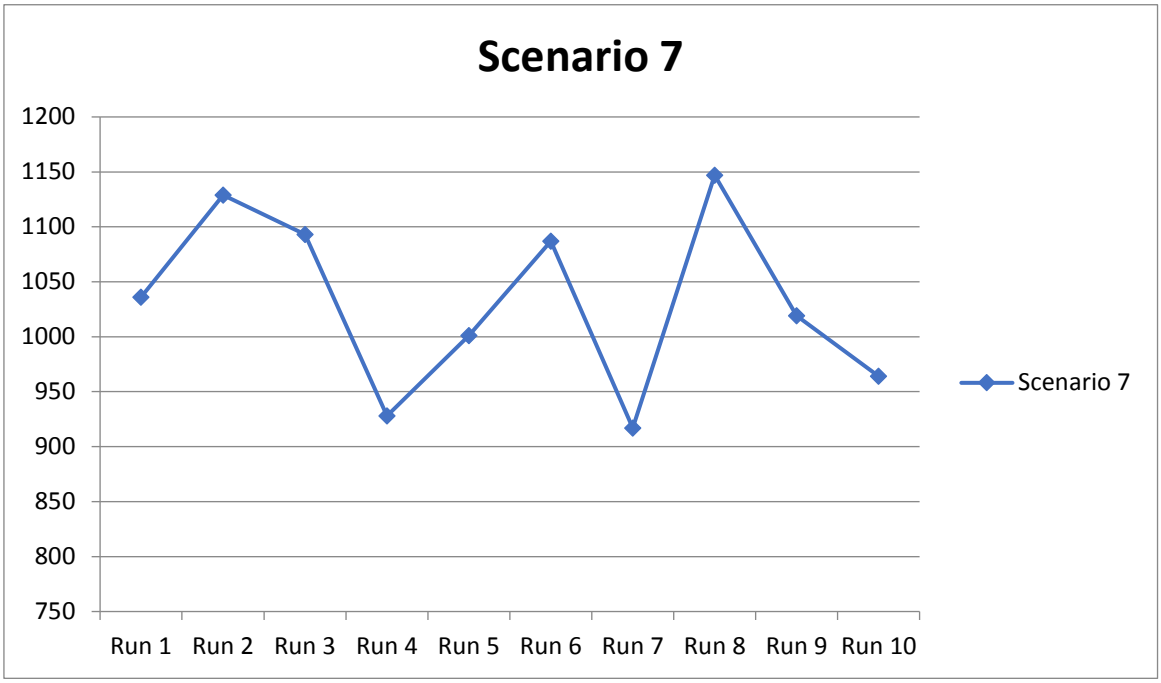


Figure 10. Generations with the best fitness values in Scenario 7 for each execution

The best solution for this scenario was the chromosome [[4, 2, 3, 6, 1], [0, 5, 7, 8, 9]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 23.468319 and was reached in the 7th run of the code. The generation needed to reach the value was 917.

3.2.8. Scenario 8

The scenario number 8 has the following situations:

1. In the 1st location, the person has a 3rd degree burn. The pain is unbearable. The risk for life is considered to be 10.
2. In the 2nd location, a person has fallen out of the window from the 4th floor. Some bones are broken. The patient is unconscious. The risk is 7.
3. The 3rd location has a person with a deep cut. The cut didn't touch any of the vital organs, but there is a lot of blood lost. The risk will be evaluated as 7.
4. The 4th location has a person with a heart attack. The person is young, and conscious. The risk will be 7.5.

5. The 5th location has 1 person. He is going through an allergic reaction. The patient is still conscious. The risk for this person's life will be evaluated as 7.
6. The 6th location will have a person with a relatively strong headache. No other symptoms were detected, so the risk for this case will be 2.
7. The 7th location is a restaurant. 6 people have been poisoned. The risk for 4 of these people is 2, the risk for the remaining 2 is 4.
8. The 9th location has 1 person. The patient is having a mild seizure. The symptoms are not life threatening so the risk for the person's life will be 5.
9. The 9th location also has 1 person. The injury is broken leg. The wound is open, so the risk increases and will be evaluated as 6.
10. The last location has 2 people. Both got hit by a car. Some of their bones are broken. The risk for this case is 5 for both people,

The scenario was again run 10 times. The results of these executions show that the fastest the algorithm reached its highest fitness value is 958, the slowest is 1191 and the average number is 1023.The comparison can be seen in Figure 11.

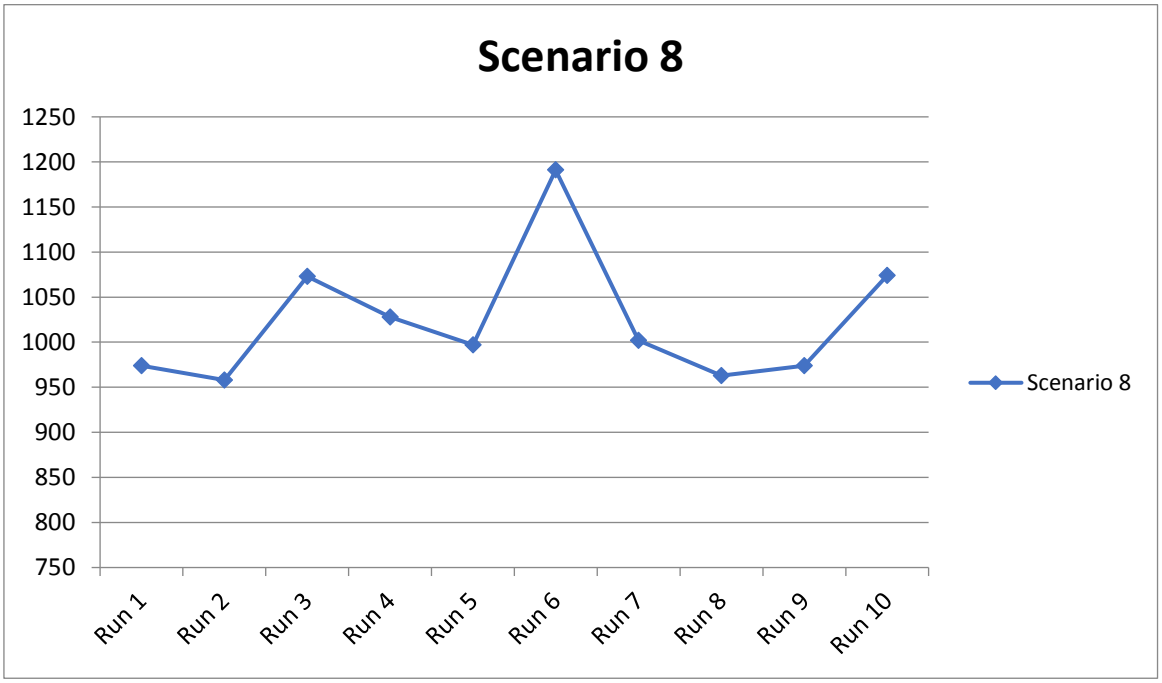


Figure 11. Generations with the best fitness values in Scenario 8 for each execution

The best solution for this scenario was the chromosome [[0, 3, 2, 9, 5], [2, 1, 6, 4, 8]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 40. 937427 and was reached in the 7th run of the code. The generation needed to reach the value was 958.

3.2.9. Scenario 9

The scenario number 9 has the following situations:

1. The 1st location has a person with a headache, nausea and dizziness. The condition has been continuing for 2 hours and is getting worse. The risk for this case will be evaluated as 5.
2. The 2nd location has a person with a heart attack. The person is young, and conscious. The risk will be 7.5.
3. The 3rd location has 1 person, an elderly person, who is going through a stroke. The person is conscious but all of the symptoms are highly defined. The risk of the injury will be evaluated as 8.
4. The 4th location has 1 person. The patient is having a panic attack. The risk is 1, because of the little damage to the patient's health.
5. In the 5th location, the person has a 3rd degree burn. The pain is unbearable. The risk for life is considered to be 10.
6. The 6th location has 1 person. The injury sustained is broken arm. The wound is close; there is no blood loss, so the risk for this case is evaluated as 2.
7. The 7th location has 1 person. She got hit by a car. Some of her bones are broken. The risk for this case is 5.
8. The 8th location has a patient with dislocated shoulder. The pain is medium. The risk will be taken as 2.

9. The 9th location will have a person with a relatively strong headache. No other symptoms were detected, so the risk for this case will be evaluated as 2.
10. In the 10th location, the person is having a stomach ache. The suspicions are that the person has eaten rotten food. Since the only symptom is the pain, the risk will be evaluated as 2.

The scenario was again run 10 times. The results of these executions show that the fastest the algorithm reached its highest fitness value is 903, the slowest is 1103 and the average number is 977. The comparison can be seen in Figure 12.

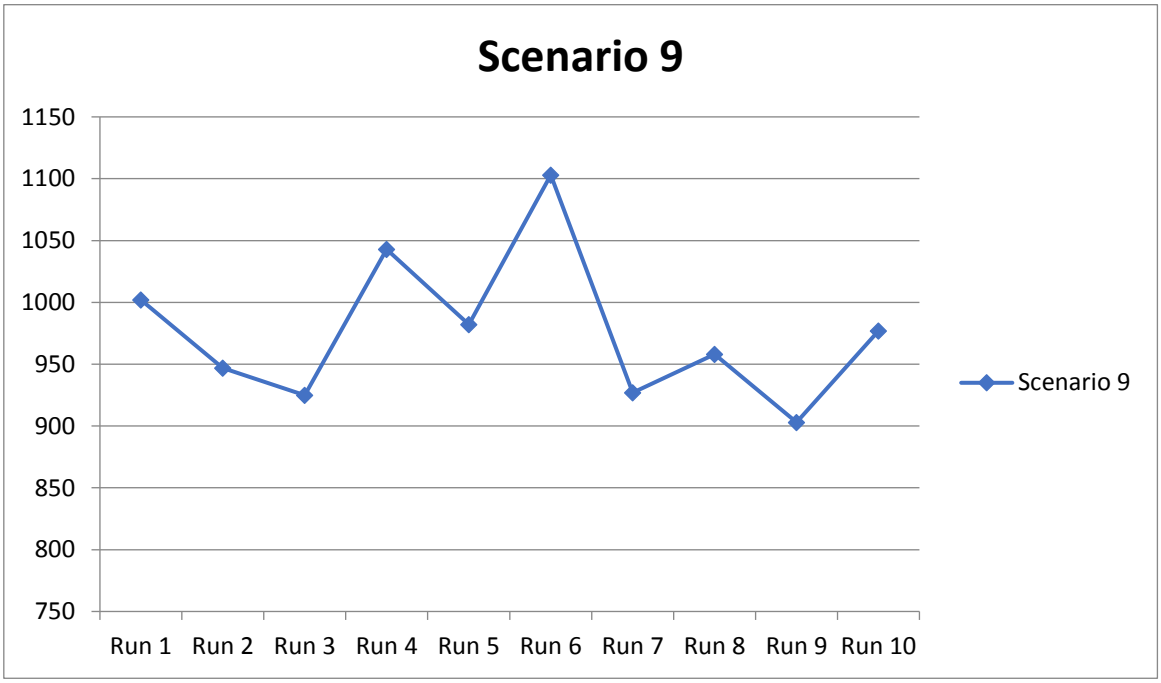


Figure 12. Generations with the best fitness values in Scenario 9 for each execution

The best solution for this scenario was the chromosome [[2, 1, 7, 8, 3], [4, 6, 0, 9, 5]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 19.672183 and was reached in the 7th run of the code. The generation needed to reach the value was 903.

3.2.10. Scenario 10

The scenario number 10 has the following situations:

1. The 1st location has 1 person, an elderly person, who is going through a stroke. The person is conscious but all of the symptoms are highly defined. The risk of the injury will be evaluated as 8.
2. The 2nd location has 1 person. The injury sustained is broken arm. The wound is close; there is no blood loss, so the risk for this case is evaluated as 2.
3. The 3rd location has 1 person. She got hit by a car. Some of her bones are broken. The risk for this case is 5.
4. The 4th location has 1 person. The patient is having a panic attack. The risk is 1, because of the little damage to the patient's health.
5. The 5th location has a person with a heart attack. The person is young, and conscious. The risk will be 7.5.
6. The 6th location will have a person with a relatively strong headache. No other symptoms were detected, so the risk for this case will be evaluated as 2.
7. The 7th location has 1 person. The injury sustained is broken arm. The wound is close; there is no blood loss, so the risk for this case is evaluated as 2.
8. The 8th location has a person with a headache, nausea and dizziness. The condition has been continuing for 2 hours and is getting worse. The risk for this case will be evaluated as 5.
9. The 9th location has 4 people. 2 of them have deep cuts, and the other 2 have 2nd degree burns. The risk for the life of the people with the cuts is 7 for both, since the cuts didn't touch any vital organs. The risk for both of the people with burns is evaluated as 5.

10. The 10th location has a person with a deep cut. The cut didn't touch any of the vital organs, but there is a lot of blood lost. The risk will be evaluated as 7.

The scenario was again run 10 times. The results of these executions show that the fastest the algorithm reached its highest fitness value is 967, the slowest is 1208 and the average number is 1056. The comparison can be seen in Figure 13.

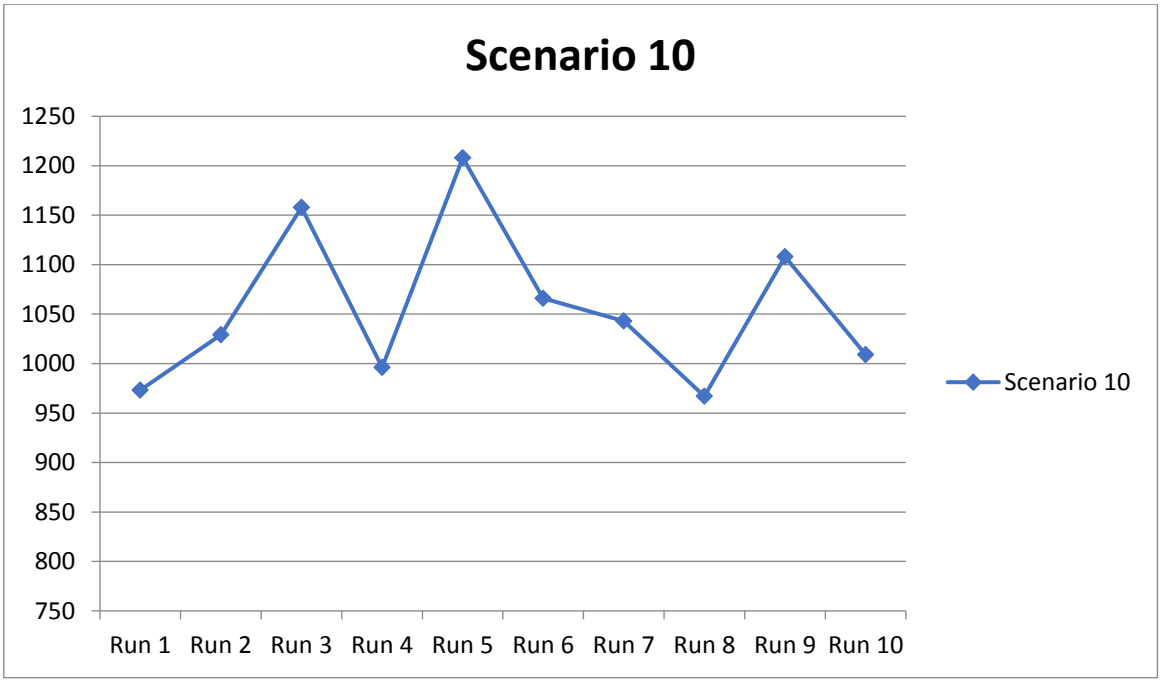


Figure 13. Generations with the best fitness values in Scenario 10 for each execution

The best solution for this scenario was the chromosome [[8, 0, 7, 6], [4, 9, 5, 2, 1, 3]]. The average fitness value of the randomly generated chromosomes was 4.398219. The highest fitness value was 29.874029 and was reached in the 8th run of the code. The generation needed to reach the value was 967.

3.2.11. Final Result

The algorithm was run 10 times for each of the scenarios. Each of the executions used 5000 generations. The result of these 10 executions for each of the Scenarios shows that the average generation, needed to get to the bestfinal fitness value in

each of the tests is 1015.8. The average for least generation for all of the scenarios was 924.9, with 873 being the smallest number and 967 being the largest.
The average for highest number of generations for all of the scenarios was 1146 with 1025 being the smallest number and 1253 being the largest.
The results are illustrated in Figure 14:

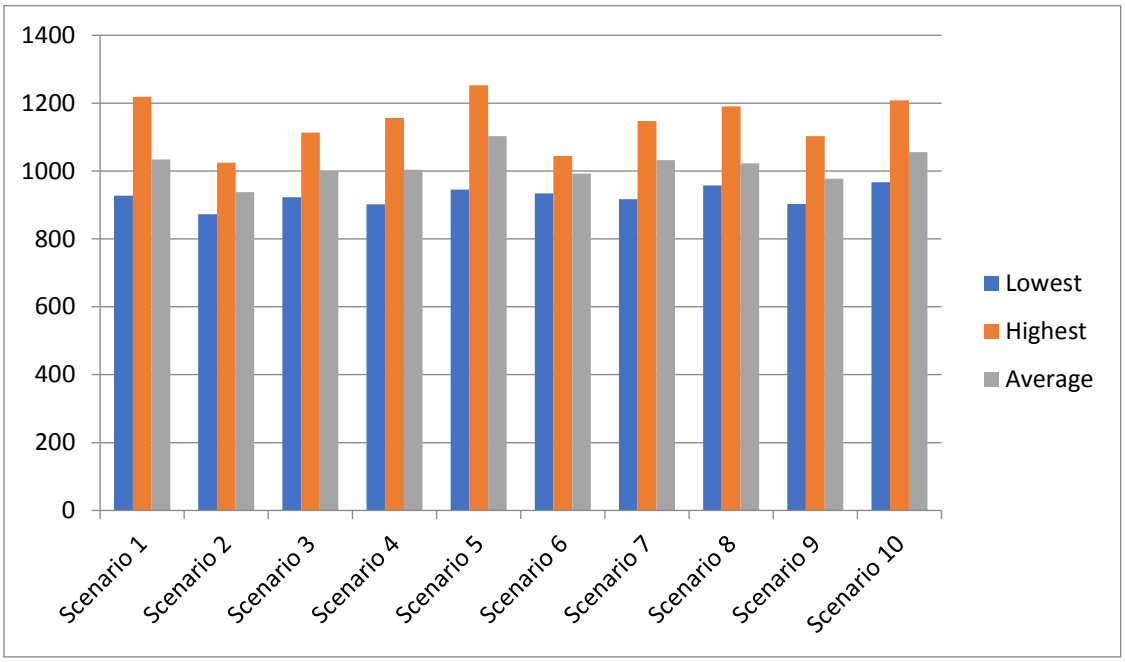


Figure 14. The lowest, average and highest generations of the executions in all of the scenarios

3.3. Comparison

The proposed algorithm was compared to the study of Ruijiu Mao, Bin Du, Dengfeng Sun, and Nan Kong [16]. The authors offered model for location-allocation optimization with mean waiting time as the aim. The study consisted of 2 parts: Estimation of mean waiting time, and using Genetic Algorithm to optimize the UAV allocation for Emergency Medical Service (EMS) purposes. For the purpose of comparing this approach to the HMOGA algorithm, the comparison was done between the HMOGA and the genetic algorithm part of the authors' research. The authors' algorithms details are described below:

The selection method was chosen to be Roulette Wheel. The main difference between the algorithms is in this phase. In this phase, the authors' study will select mostly chromosomes with higher fitness values. Even though this method is one of the most popular ones, choosing chromosomes which have higher fitness chromosomes does not guarantee that the child will also have high fitness value. The HMOGA on the other hand, chooses chromosomes with varying fitness values. Although this also does not guarantee child chromosomes with high fitness values, this method enables more chromosomes to be involved into the process.

The next step is crossover. The crossover method chosen for the authors' study is uniform crossover. The method was described in the previous chapters.

After performing crossover, mutation was performed. This phase is the same for both of the algorithms.

The previously described 10 scenarios were tested on the authors' genetic algorithm method. The results are listed below:

The average iterations needed to get to the highest fitness function for the first scenario was 1202. The lowest number was 1014, and the highest one was 1377. The results are illustrated in Figure 15, along with the results of HMOGA from Figure 4.

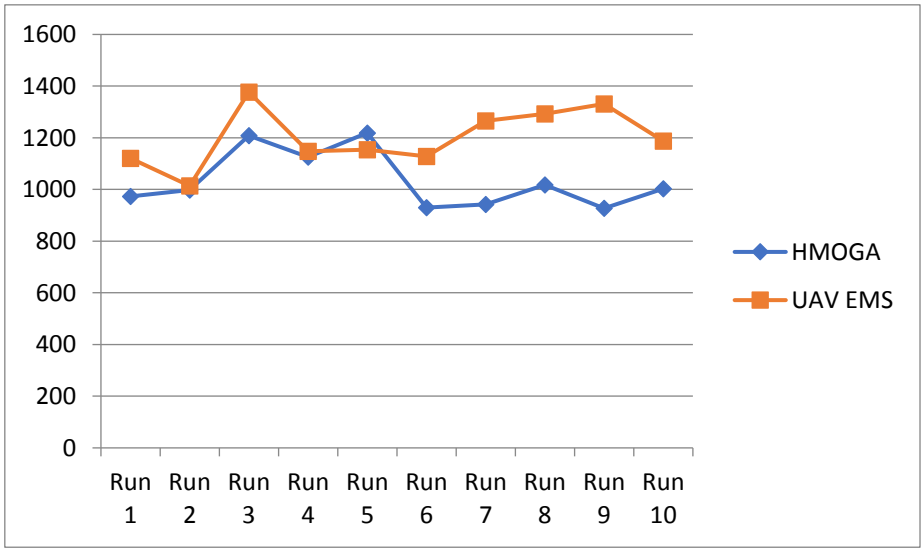


Figure 15. Comparison of average generation with the best fitness function in Scenario 1 of HMOGA and UAV EMS approach

The average for the second scenario was 1190, the lowest – 983, the highest – 1306. The results are illustrated in Figure 16, along with the results of HMOGA from Figure 5.

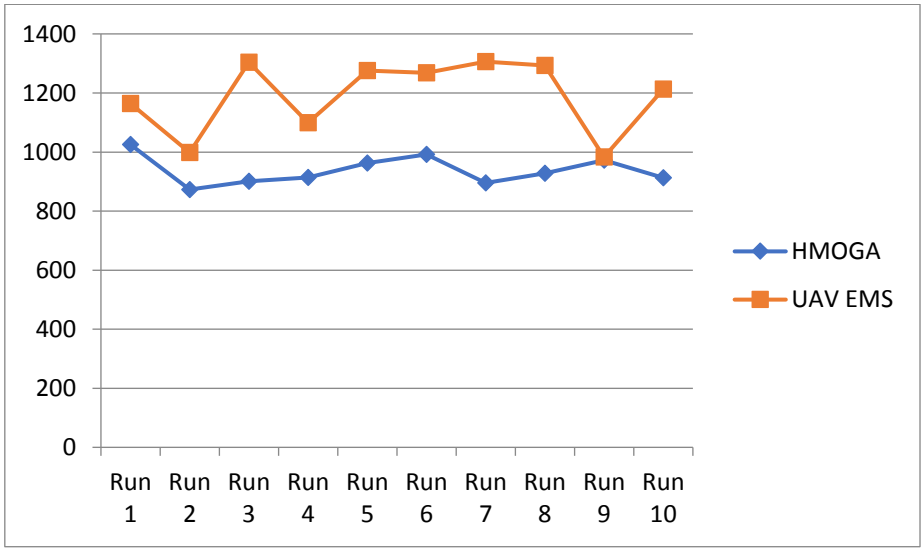


Figure 16. Comparison of average generation with the best fitness function in Scenario 2 of HMOGA and UAV EMS approach

The average for the third scenario was 1308, the lowest – 1105, the highest – 1485. The results are illustrated in Figure 17, along with the results of HMOGA from Figure 6.

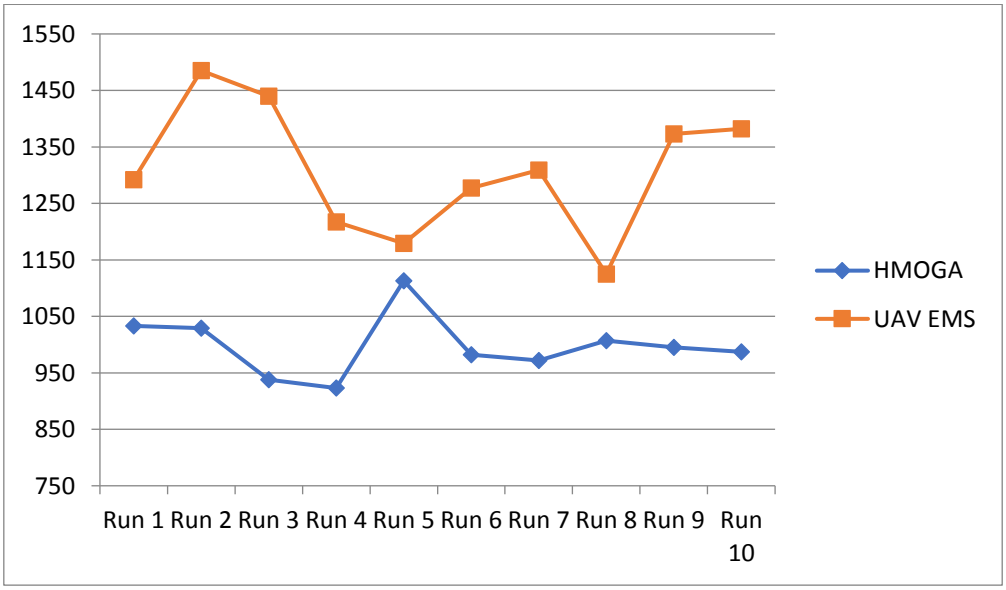


Figure 17. Comparison of average generation with the best fitness function in Scenario 3 of HMOGA and UAV EMS approach

The average for the fourth scenario was 1107, the lowest – 1002, the highest – 1296. The results are illustrated in Figure 18, along with the results of HMOGA from Figure 7.

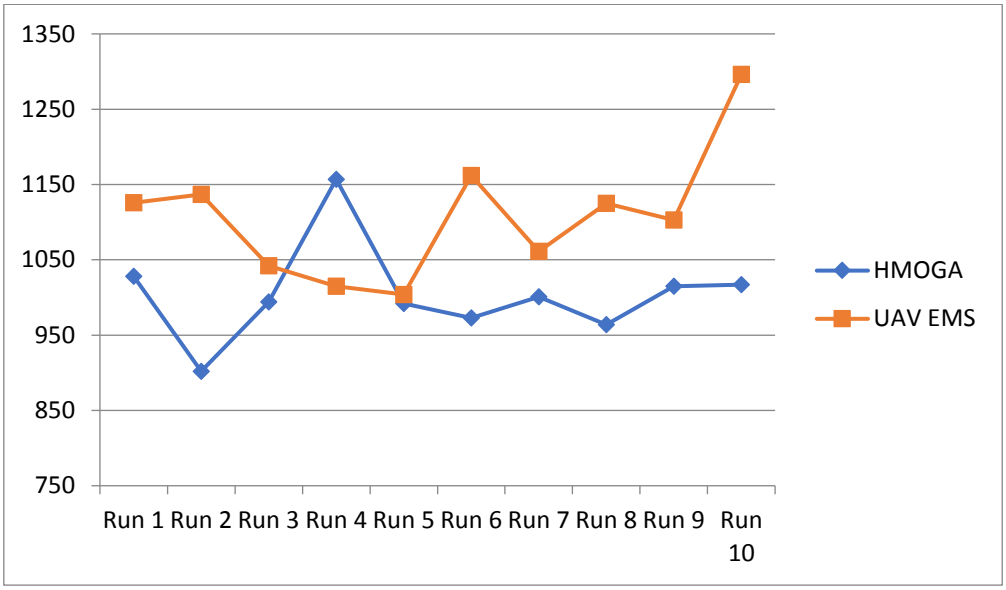


Figure 18. Comparison of average generation with the best fitness function in Scenario 4 of HMOGA and UAV EMS approach

The average for the fifth scenario was 1093, the lowest – 996, the highest – 1254. The results are illustrated in Figure 19, along with the results of HMOGA from Figure 8.

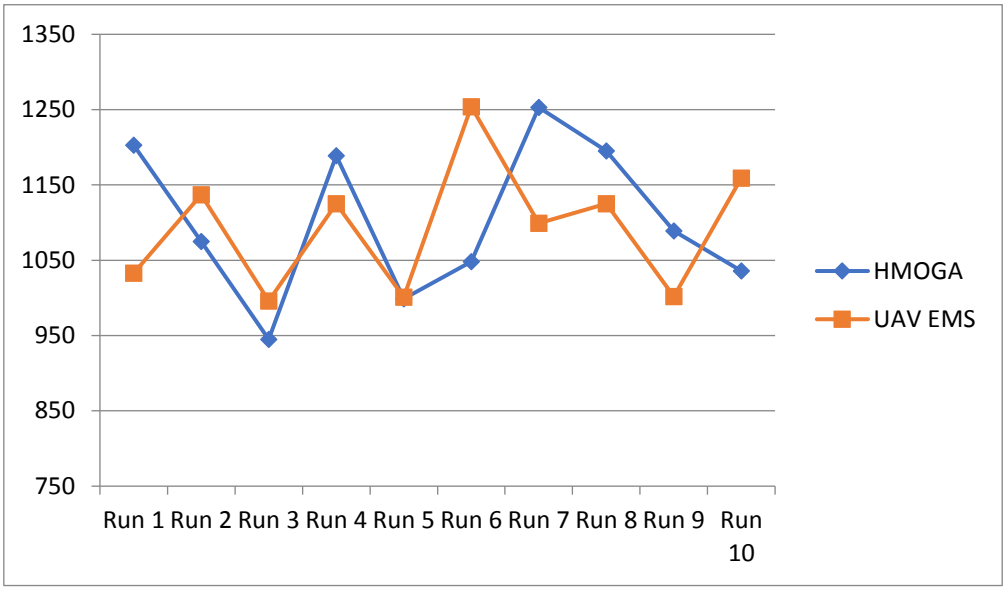


Figure 19. Comparison of average generation with the best fitness function in Scenario 5 of HMOGA and UAV EMS approach

The average for the sixth scenario was 1031, the lowest – 1008, the highest – 1194. The results are illustrated in Figure 20, along with the results of HMOGA from Figure 9.

Figure 20. Comparison of average generation with the best fitness function in Scenario 6 of HMOGA and UAV EMS approach

The average for the seventh scenario was 1007, the lowest – 961, the highest – 1295. The results are illustrated in Figure 21, along with the results of HMOGA from Figure 10.

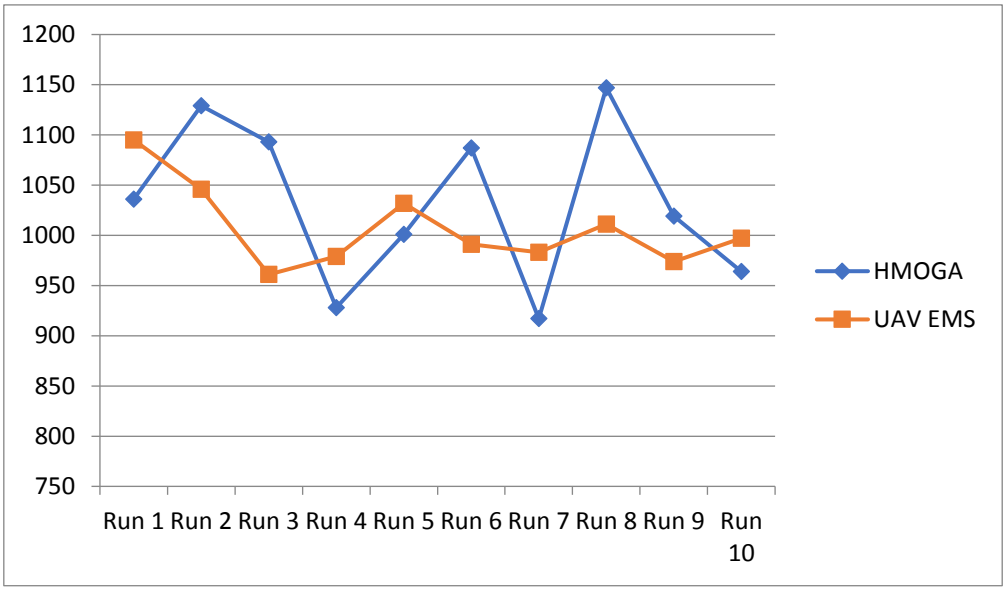


Figure 21. Comparison of average generation with the best fitness function in Scenario 7 of HMOGA and UAV EMS approach

The average for the eighth scenario was 1324, the lowest – 1097, the highest – 1474. The results are illustrated in Figure 22, along with the results of HMOGA from Figure 11.

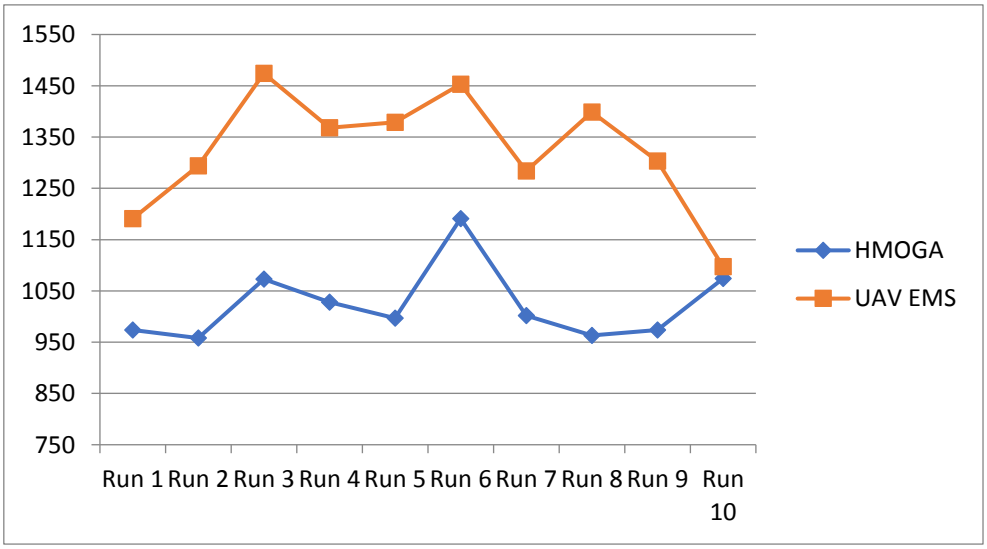


Figure 22. Comparison of average generation with the best fitness function in Scenario 8 of HMOGA and UAV EMS approach

The average for the ninth scenario was 1109, the lowest – 1015, the highest – 1378. The results are illustrated in Figure 23.

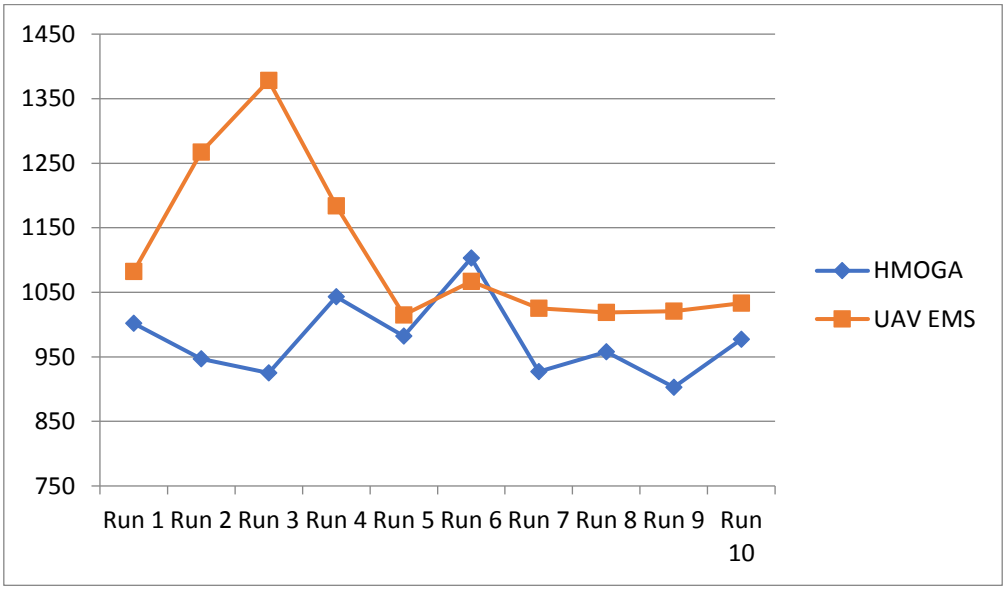


Figure 23. Comparison of average generation with the best fitness function in Scenario 9 of HMOGA and UAV EMS approach

The average for the last scenario was 1034, the lowest – 979, the highest – 1172. The results are illustrated in Figure 23, along with the results of HMOGA from Figure 12.

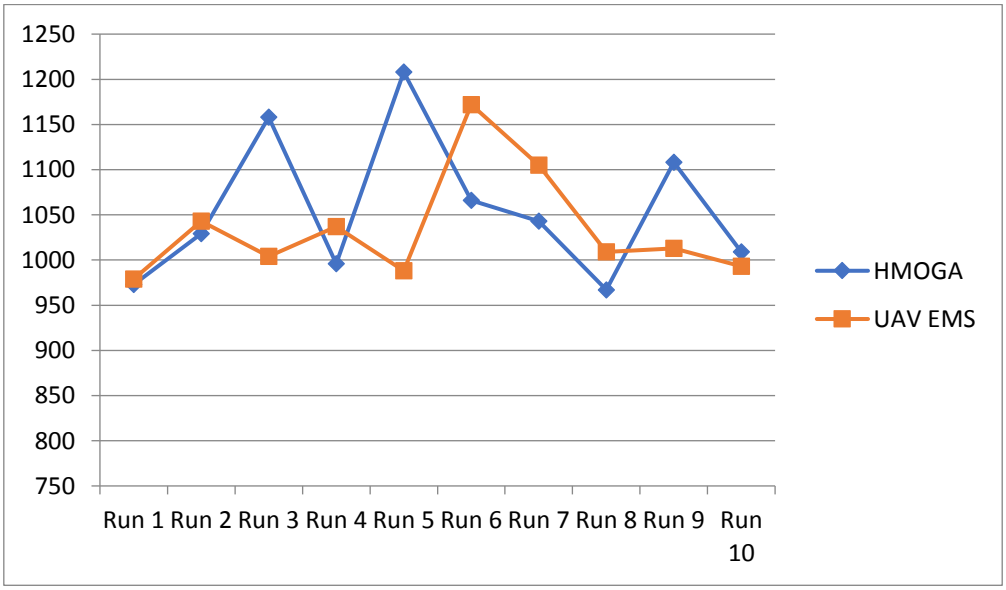


Figure 24. Comparison of average generation with the best fitness function in Scenario 10 of HMOGA and UAV EMS approach

The total average for this approach is 1140.5, the lowest is 961, and the highest is 1485. The results are illustrated in Figure 25.

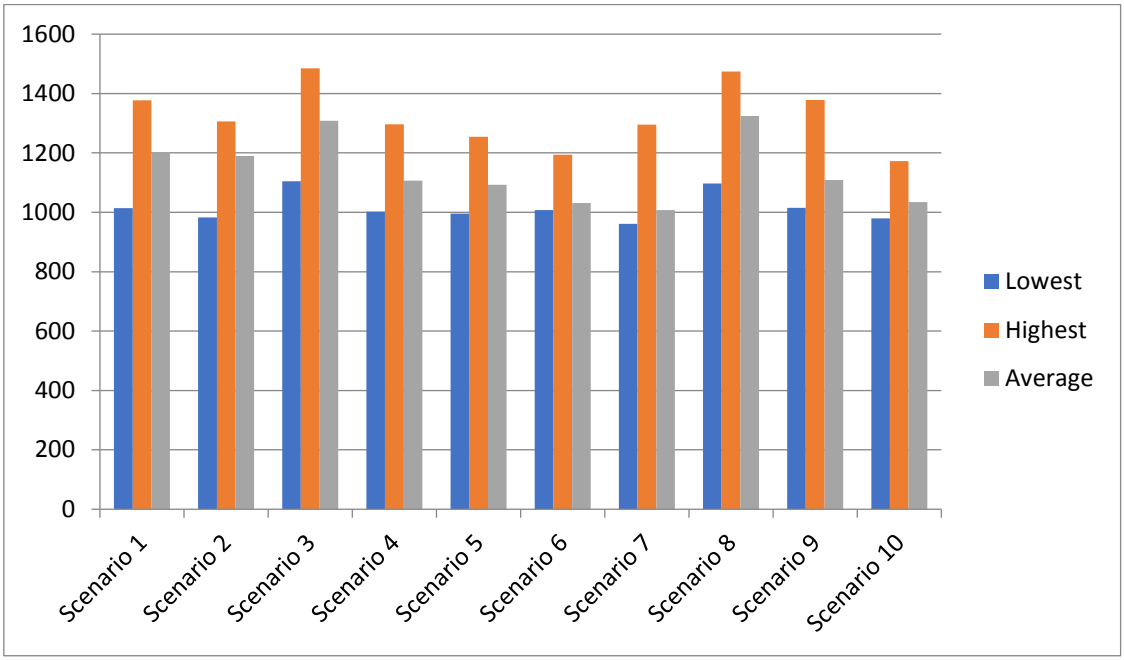


Figure 25. The lowest, average and highest generation counts of the executions in all of the scenarios in UAV EMS approach

When comparing these results to the results of the proposed HMOGA algorithm, we can see that out of 10 different scenarios, 7 reached their best solution faster with the Genetic Algorithm with KNN used to select the parent chromosomes. The average generation for HMOGA to reach the best fitness value is 1015.9, whereas for UAV EMS approach it is 1140.5. This shows that the HMOGA finds in 12.28% less generations, and due to that, faster. Also, the HMOGA considers more variables, like number of saved people and the risk of each injury. Figure 26 demonstrates the average number of iterations for each scenario by HMOGA and the UAV-based EMS system:

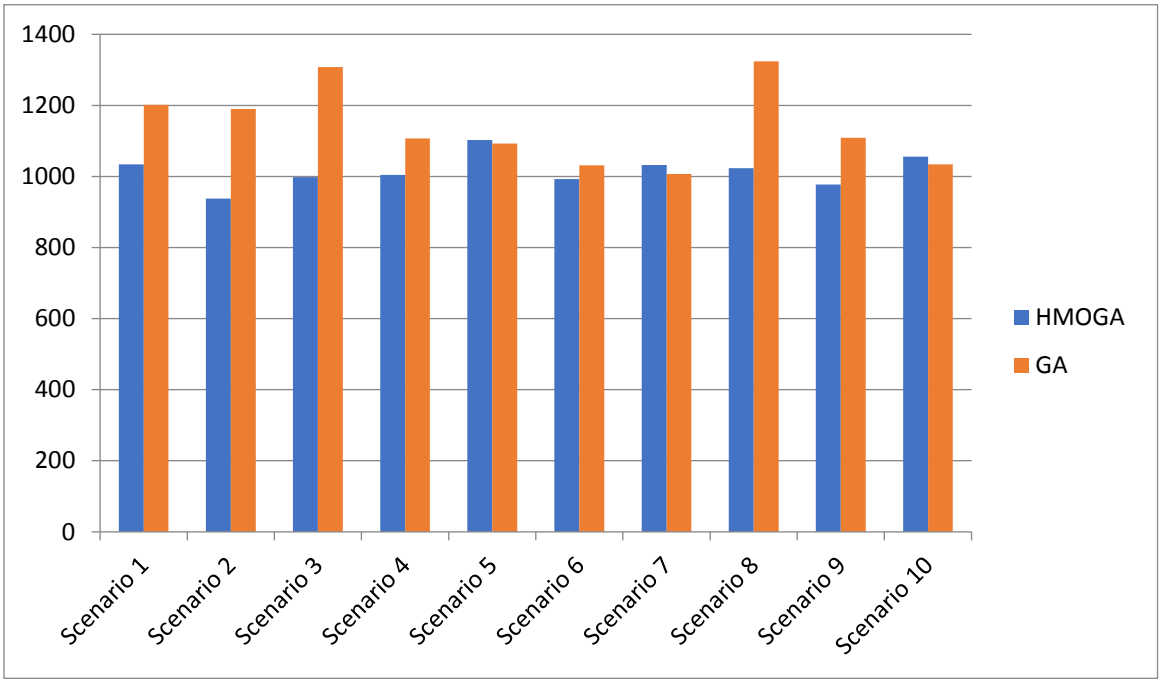


Figure 26. Generations where the fitness value reached it's highest

4. Conclusion

In this study I proposed an algorithm that combines two techniques: Genetic algorithm and K-nearest neighbors algorithm. The latter was used in the selection phase of genetic algorithm to cause a diversion of the selected parent chromosomes based on their classifications. After applying KNN to the randomly generated solutions, basic genetic algorithm operations were performed, such as crossover and mutation. The HMOGA algorithm was then compared to a previously presented approach and in 70% of the cases was proven to be a faster method to reach the optimal solution. By the average time of generations, the Hybrid Multi-Objective Genetic Algorithm was 12.28% faster. In the future, the changes can include different types of crossover and mutation operations, which will be more suitable for the problem, given the relativeness to the Traveling Salesman Problem.

5. References

1. Abd, G., Mahmoud, A. & El-Horbarty, El. (2014). A Comparative Study of Meta-heuristic Algorithms for Solving Quadratic Assignment Problem. International Journal of Advanced Computer Science and Applications. 5. 10.14569/IJACSA.2014.050101.
2. Abdoun, O., Abouchabaka, J., & Tajani, C. (2012). Analyzing the performance of mutation operators to solve the travelling salesman problem. arXiv preprint arXiv:1203.3099.
3. Ai, B., Li, B., Gao, S., Xu, J., & Shang, H. (2019). An intelligent decision algorithm for the generation of maritime search and rescue emergency response plans. IEEE Access, 7, 155835-155850.
4. Alabsi, F., & Naoum, R. (2012). Comparison of selection methods and crossover operations using steady state genetic based intrusion detection system. Journal of Emerging Trends in Computing and Information Sciences, 3(7), 1053-1058.
5. Alzyadat, T., Yamin, M., & Chetty, G. (2020). Genetic algorithms for the travelling salesman problem: a crossover comparison. International Journal of Information Technology, 12(1), 209-213.
6. Arowolo, M. O., Adebiyi, M., Adebiyi, A., & Okesola, O. (2020, March). PCA model for RNA-Seq malaria vector data classification using KNN and decision tree algorithm. In 2020 international conference in mathematics, computer engineering and computer science (ICMCECS) (pp. 1-8). IEEE.
7. Chicano, F., Sutton, A. M., Whitley, L. D., & Alba, E. (2015). Fitness probability distribution of bit-flip mutation. Evolutionary Computation, 23(2), 217-248.
8. Crişan, G. C., Pintea, C. M., & Palade, V. (2017). Emergency management using geographic information systems: application to the first romanian

traveling salesman problem instance. Knowledge and Information Systems, 50(1), 265-285.

9. Dong, H., Li, T., Ding, R., & Sun, J. (2018). A novel hybrid genetic algorithm with granular information for feature selection and optimization. Applied Soft Computing, 65, 33-46.
10. Fan, Y., Bai, J., Lei, X., Zhang, Y., Zhang, B., Li, K. C., & Tan, G. (2020). Privacy preserving based logistic regression on big data. Journal of Network and Computer Applications, 171, 102769.
11. Fernandez, B., Fanggidae, A., Pandie, E. S. Y., & Mauko, A. Y. (2021, September). Travelling salesman problem: Greedy single point crossover in ordinal representation. In Journal of Physics: Conference Series (Vol. 2017, No. 1, p. 012012). IOP Publishing.
12. Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80(5), 8091-8126.
13. Yadav, S. L., & Sohal, A. (2017). Comparative study of different selection techniques in genetic algorithm. International Journal of Engineering, Science and Mathematics, 6(3), 174-180.
14. Lathuilière, S., Mesejo, P., Alameda-Pineda, X., & Horaud, R. (2019). A comprehensive analysis of deep regression. IEEE transactions on pattern analysis and machine intelligence, 42(9), 2065-2081.
15. Leite, R., Brazdil, P., & Vanschoren, J. (2012, July). Selecting classification algorithms with active testing. In International workshop on machine learning and data mining in pattern recognition (pp. 117-131). Springer, Berlin, Heidelberg.
16. Mao, R., Du, B., Sun, D., & Kong, N. (2019, August). Optimizing a UAV-based emergency medical service network for trauma injury patients. In

2019 IEEE 15th International Conference on Automation Science and Engineering (CASE) (pp. 721-726). IEEE.

17. Mirjalili, S. (2018). Genetic Algorithm. Evolutionary Algorithms and Neural Networks, 43–55.
18. Mohammed, M. A., Abd Ghani, M. K., Hamed, R. I., Mostafa, S. A., Ibrahim, D. A., Jameel, H. K., & Alallah, A. H. (2017). Solving vehicle routing problem by using improved K-nearest neighbor algorithm for best solution. Journal of Computational Science, 21, 232-240.
19. Pal, R., Srivastava, M., Rani, S., & Kumar, N. (2019, April). Genetic Algorithm towards Flood Avoidance in Android Application. In 2019 International Conference on Automation, Computational and Technology Management (ICACTM) (pp. 357-360). IEEE.
20. Pallin, M., Rashid, J., & Ögren, P. (2021, October). A decentralized asynchronous collaborative genetic algorithm for heterogeneous multi-agent search and rescue problems. In 2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) (pp. 1-8). IEEE.
21. Rashid, M. H., & Mosteiro, M. A. (2017, December). A Greedy-Genetic Local-Search Heuristic for the Traveling Salesman Problem. In 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC) (pp. 868-872). IEEE.
22. Ryerkerk, M. L., Averill, R. C., Deb, K., & Goodman, E. D. (2017). Solving metameric variable-length optimization problems using genetic algorithms. Genetic Programming and Evolvable Machines, 18(2), 247-277.
23. Santafe, G., Inza, I., & Lozano, J. A. (2015). Dealing with the evaluation of supervised classification algorithms. Artificial Intelligence Review, 44(4), 467–508.

24. Tripathy, S. P., Tulshyan, A., Kar, S., & Pal, T. (2017). A metameric genetic algorithm with new operator for covering salesman problem with full coverage. Int J Control Theory Appl, 10(7), 245-252.
25. Umbarkar, A. J., & Sheth, P. D. (2015). Crossover operators in genetic algorithms: a review. ICTACT journal on soft computing, 6(1).
26. Yang, F. J. (2018, December). An implementation of naive bayes classifier. In 2018 International conference on computational science and computational intelligence (CSCI) (pp. 301-306). IEEE.