Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 225 (2023) 3536–3545

Procedia Computer Science

www.elsevier.com/locate/procedia

27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2023)

A Genetic Algorithm for Optimizing Mobile Stroke Unit Deployment

Muhammad Adil Abid[a,*], Saeid Amouzad Mahdiraji[a], Fabian Lorig[a,b], Johan Holmgren[a], Radu-Casian Mihailescu[a], Jesper Petersson[c,d]

[a] *Department of Computer Science and Media Technology, Malmö University, 21119 Malmö, Sweden*
[b] *Internet of Things and People Research Center, Malmö University, Sweden*
[c] *Department of Health Care Management, Region Skåne, 21428 Malmö, Sweden*
[d] *Department of Neurology, Lund University, 22242 Malmö, Sweden*

Abstract

A mobile stroke unit (MSU) is an advanced ambulance equipped with specialized technology and trained healthcare personnel to provide on-site diagnosis and treatment for stroke patients. Providing efficient access to healthcare (in a viable way) requires optimizing the placement of MSUs. In this study, we propose a time-efficient method based on a genetic algorithm (GA) to find the most suitable ambulance sites for the placement of MSUs (given the number of MSUs and a set of potential sites). We designed an efficient encoding scheme for the input data (the number of MSUs and potential sites) and developed custom selection, crossover, and mutation operators that are tailored according to the characteristics of the MSU allocation problem. We present a case study on the Southern Healthcare Region in Sweden to demonstrate the generality and robustness of our proposed GA method. Particularly, we demonstrate our method's flexibility and adaptability through a series of experiments across multiple settings. For the considered scenario, our proposed method outperforms the exhaustive search method by finding the best locations within 0.16, 1.44, and 10.09 minutes in the deployment of three MSUs, four MSUs, and five MSUs, resulting in 8.75x, 16.36x, and 24.77x faster performance, respectively. Furthermore, we validate the method's robustness by iterating GA multiple times and reporting its average fitness score (performance convergence). In addition, we show the effectiveness of our method by evaluating key hyperparameters, that is, population size, mutation rate, and the number of generations.

Keywords: genetic algorithm; mobile stroke unit (MSU); optimization; healthcare; time to treatment

* Muhammad Adil Abid. Tel.: +46-070-014-2616
E-mail address: muhammad.adil-abid@mau.se

10.1016/j.procs.2023.10.349

1. Introduction

A stroke is a serious life-threatening medical condition that occurs when a blockage of the blood vessels either interrupts or reduces the blood supply to the brain. Without immediate treatment, the patient has a low chance of a successful recovery. Stroke affects one in six people globally during their lifetime, with 15 million new cases and 5.8 million deaths annually. In Sweden, over 21,000 cases occur yearly, with 3,900 in the Southern Healthcare Region (SHR) [8, 9]. Stroke also leads to long-term disability and financial hardship for individuals and families. It is generally agreed that patients who receive treatment within an hour are substantially more likely to recover than those who receive treatment later. However, immediate treatment of stroke patients is challenging due to logistical constraints and difficulty in obtaining a correct stroke diagnosis.

There are three primary types of stroke: ischemic, hemorrhagic, and transient ischemic attack. Ischemic strokes occur when one or more clots restrict blood flow to the brain, and they can be treated with thrombolysis and, in some cases, thrombectomy. Hemorrhagic strokes occur when a blood vessel in the brain ruptures, and blood pressure-lowering therapy is recommended as soon as possible. Transient ischemic attacks happen when blood flow to the brain is temporarily blocked by a clot, but brain function is able to fully recover [5]. The main barrier to immediate treatment is that different types of stroke require different treatments. Patients suffering from ischemic strokes should typically be treated with drugs that dissolve blood clots. However, this form of treatment should never be provided to patients suffering from bleeding (i.e., hemorrhagic stroke), as it would result in the patient's death. Since patients with different types of strokes often have similar symptoms, a brain imaging technique called computed tomography (CT) is needed for diagnosis [6].

In this work, we study the use of so-called Mobile Stroke Units (MSUs) in addition to standard ambulances. An MSU is a specialized ambulance equipped with a CT scanner, allowing the ambulance staff to diagnose stroke patients and give intravenous stroke treatment while the patient is still in the ambulance. As a result, the treatment time can typically be cut down, at least corresponding to the time needed to transport the patient to an acute hospital. However, regions typically only have a few MSUs in a region since the operational cost for an MSU is high; thus, it is important to strategically place them in order to provide maximum benefits for the patients in the region.

In our companion study, Mahdiraji et al. [7] use an exhaustive search (ES) to solve the optimization problem of allocating MSUs. Although an ES is straightforward and provides a solution, it is computationally expensive and infeasible for greater search spaces due to the traversing through all candidate solutions. In fact, ES becomes unmanageably slow, impractical, and unusable for larger problem instances. Therefore, we found it worthwhile to explore other optimization techniques that can efficiently determine the suitable solution for the placement of MSU locations. One such technique is the use of a genetic algorithm (GA), which can provide solutions without explicitly evaluating all possible combinations of MSU locations, thus reducing computation time. To efficiently solve optimization problems, GA has received a lot of attention in multiple domains, including portfolio optimization, vehicle routing, and facility location [4]. Previous research has also established the effectiveness of GA in solving ambulance dispatching and allocation types of optimization problems [1, 2, 3]. Given the efficiency of GA in solving optimization problems, particularly for ambulance dispatching and allocation, it has inspired us to propose a GA method capable of efficiently searching for the optimal locations for a number of MSUs.

The current paper addresses the optimization problem of allocating MSUs in a geographic region using GA. We carried out a study aiming to develop a robust and efficient method to strategically place a number of MSUs at different potential sites within a specific region to minimize time to treatment. We mainly focused on an efficient encoding scheme for the input data (the number of MSUs and potential sites) and the design of the custom selection, crossover, and mutation operators in accordance with the nature of the MSU allocation problem. We evaluated our model's performance by comparing it with a brute force technique (i.e., ES); see the companion paper by Amouzad Mahdiraji et al.[7], which provides a scenario study for Sweden's SHR. To demonstrate the feasibility and adaptability of our proposed method, we consider multiple settings (i.e., the deployment of three, four, and five MSUs) on 39 potential sites in the SHR. We compare our proposed method's performance in terms of time complexity. In addition, we visualize the convergence of the optimization process, highlighting the evolution of the best-found MSUs location over time and generations.

The key contributions are as follows:

1. Proposing a GA for finding the best MSU locations.
2. Designing input encoding procedure and GA operators.

3. Performing systematic experimentation across multiple settings (3MSUs, 4MSUs, and 5MSUs) to assess the robustness of the proposed algorithm.

The rest of the paper is structured in the following way: Section 2 reviews related work and Section 3 introduces the formal definition of the MSU optimization problem. Section 4 presents the proposed GA. Our case study, as well as the results from the experiments, are presented in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

In emergency medical services (EMS), several studies employed a GA for optimal ambulance location, relocation, and fleet allocation problems. Zhen et al. [10] propose a simulation-optimization framework to assess the operational performance of ambulance deployment plans. A GA, integrated into the framework, is used to identify a near-optimal solution, that is, an ambulance deployment decision, in the solution space. The simulation model takes a potential solution of the considered optimization problem and assesses the solution's performance in a stochastic environment. Liu et al. [11] present a double standard model, integrated with a GA, to assess emergency vehicle service fleet allocation from their locations to incident sites to maximize service coverage standards. Further, McCormack and Coates [12] combined a GA with a simulation model to optimize ambulance fleet allocation and station location to maximize the total expected survival probability of different patient classes. Tili et al. [13] explore the use of a GA to solve the ambulance routing problem in two scenarios: a simple ambulance routing problem and an open one. Their model aims to serve a higher number of patients by using a fixed number of resources and minimizing the total travel distance for the same number of requests. Zaheeruddin and Gupta [14] propose a novel optimization approach using hybrid particle swarm optimization and a genetic algorithm for ambulance location problems, aiming to reduce the ambulance response time.

Moreover, some papers aim to find the optimal locations for MSUs in particular regions for improved prehospital stroke care. Phan et al. [15] employ Google Maps to identify the optimal location for an MSU in Sydney. Rhudy Jr. et al. [16] propose a geospatial analysis of the distribution of an MSU to optimize service delivery for stroke patients in the city of Memphis. Dahllöf et al. [17] present an expected value optimization approach to identify the optimal location for an MSU in the Skåne County of Sweden, exploring the impact of the optimal placing for an MSU for the inhabitants of urban or rural areas. Recently, Amouzad Mahdiraji et al. [7] use ES to deploy MSUs for prehospital stroke care in the SHR. ES systematically enumerates all possible MSU locations for the solution and checks whether each solution satisfies the problem's statement, and evaluates how good it is using an objective function. However, when the number of possible MSU locations grows, the ES generates a search space that expands exponentially. Hence, the computational cost associated with the ES is often infeasible in terms of time. In order to address the limitation of time complexity, we develop a method based on an evolutionary approach that efficiently finds suitable locations to place MSUs.

3. MSU Optimization Problem

Our study considers the problem of allocating a fixed number of MSUs to a set of existing ambulance stations in a geographic region. This problem can be formulated as a mathematical optimization problem, where an optimization problem, as described below, is to place MSUs for the inhabitants of the SHR.

We let I denote the set of existing ambulance stations located in the SHR region and N the number of MSUs to allocate. We assume that each ambulance station always has at least one available regular ambulance. We further assume that the considered geographic region R is divided into a set of smaller subregions r; all patients in subregion $r \in R$ are assumed to be located in the same location, for example, the centroid of r.

We let t_{ir}^{RA} denote the expected time to treatment for a patient located in subregion r if it is served by an ambulance in station $i \in I$, and t_{ir}^{MSU} denotes the expected time to treatment for a patient located in r if it is served by an MSU located in station i. Thus, the shortest expected time to treatment for a patient located in the subregion $r \in R$ when it is served by a regular ambulance is $t_r^{RA} = \min_{i \in I}\{t_{ir}^{RA}\}$. Please note that the t_r^{RA}:s ($r \in R$) and the t_{ir}^{MSU}:s ($r \in R$, $i \in I$) can be pre-calculated and are therefore parameters in the optimization model.

To formulate the objective function of our optimization problem, we also need to introduce Q_r ($r \in R$), which denotes the expected share of the stroke cases in subregion r. We introduce our decision variables $x_i \in \{0, 1\}$, ($i \in I$) so that

$$x_i = \begin{cases} 1 & \text{if an MSU is placed in location } i \\ 0 & \text{Otherwise.} \end{cases}$$

Using the x_i:s, we can calculate the shortest expected time to treatment for a patient in subregion r when it is served by an MSU as

$$t_r^{MSU} = \min_{i \in I}\{t_{ir}^{MSU} + (1 - x_i) \cdot M\}, \tag{1}$$

where $M > 0$ is a sufficiently large constant value. For example, M can be set to the value of the largest expected time to treatment for any subregion r and any ambulance location i. Importantly, this equation will assign expected time to treatment for those stations where no MSU is allocated to such a large value that they will not be considered by the model.

The objective function of the optimization model, which also corresponds to the fitness function of our genetic algorithm is the so-called weighted time to treatment over all subregions $r \in R$ formulated as:

$$\min z = \sum_{r \in R} Q_r \cdot \min\{t_r^{RA}, t_r^{MSU}\}, \tag{2}$$

where the values of the decision variables ($x_i, i \in I$) are implicitly captured in the calculations of the t_r^{MSU}:s (see also Eq. 1). The MSU allocation, that is, the assignment of the x_i values is subject to the constraint

$$\sum_{i \in I} x_i = N, \tag{3}$$

which forces the optimization algorithm to set exactly N of the x_i variables to 1, corresponding to locating N MSUs in the region under consideration. We refer the reader to our companion article by Amouzad Mahdiraji et al. [7] for a more detailed description of the objective function.

4. Genetic Algorithm

The description of the method is divided into two sections. First, we discuss the GA's key components (i.e., input encoding, mutation, crossover operators, and fitness function). Second, we discuss the complete process of the GA (i.e., initialization, selection, reproduction, and termination criteria).

4.1. Genetic Algorithm Key Components

4.1.1. Input Encoding

Considering N MSUs and I sites where $N<I$, we need to place N MSUs among I sites. For example, we have $I = 39$ ambulance sites, and we need to place $N = 3$ MSUs at a time; consequently, the combination of the possible locations for three MSUs' placements can be any of the ones in the set: $S = (1, 2, 3), (1, 2, 4), (1, 2, 5), \cdots, (37, 38, 39)$. For instance, the location $(1, 2, 5)$ means that MSUs are placed on the 1st, 2nd, and 5th sites.

To encode our problem, we chose to use a binary encoding scheme to represent the chromosomes (MSUs' locations) in terms of bits (0s and 1s). Binary encodings can provide significant advantages in optimization problems, such as easy-to-manipulate individual bits in the representation. For our problem, one of the key benefits of a binary encoding is that it allows for the efficient representation of multiple variables within a single chromosome. Specifically, this approach enables the representation of both the number of MSUs (represented by the number of 1s) and potential sites (represented by the number of bits) within the same chromosome. This encoding simplifies the representation of the solution by using a fixed-length chromosome, where each bit corresponds to a potential site for an MSU. Moreover, it provides a standard way to apply genetic operators such as crossover and mutation to the chromosome during the optimization process. Following the case of the three MSUs settings, each chromosome consists of three MSUs represented using a 39 bits encoding. Therefore, we turn on three bits according to the locations of the MSUs, for example, $(1, 2, 5) \rightarrow$ (110010000000000000000000000000000000000). Due to the binary encoding, we do not need to perform any operations in the subsequent steps (crossover and mutation), which enables our model to manipulate the representation efficiently and to implement crossover and mutation operations faster. In this example, we suppose three MSUs' placement among 39 potential sites, resulting in a search space of 9,139 possible solutions. Generally, we can calculate the total number of solutions in the search space as

$$\binom{I}{N} = \frac{I!}{N!(I-N)!} \tag{4}$$

4.1.2. Mutation

The mutation is a genetic operator used to maintain genetic diversity from one generation of a population of GA chromosomes to the next. The goal is to randomly change a bit (representing a location) in one chromosome, increasing the population's diversity. To achieve this, we use a hyper-parameter known as the mutation rate, which determines the probability of a mutation occurring in a location of an MSU (i.e., a single bit of the chromosome). The mutation rate ranges from 0 to 1, inclusive. To perform a mutation on a chromosome, we go through each MSU location (bit) among I potential locations (bits) and flip it (from $0 \rightarrow 1$ or $1 \rightarrow 0$) based on the mutation rate to generate the final output. Typically, $1/L$ is suggested as a good mutation rate, where L is the length of the bitstring (in our case, the number of potential sites).

In our binary string, each "1" bit denotes the presence of an MSU in a particular location in the 39 bits string, and maintaining the appropriate number of "1" bits (representing the number of MSUs) during the mutation process is required, following Eq. 3. Our designed mutation strategy involves identifying the positions of all "1" bits in the input string and calculating the mutation rate per "1" bit. We stop this process as soon as we have "flipped" one bit. Each "1" bit is evaluated individually, and if a mutation occurs, the process ensures that the number of "1" bits in the string remains the same. Specifically, the mutated bit is swapped with a randomly selected "0" bit to maintain the same number of MSUs in the offspring. To illustrate our mutation operator working, we consider MSU location combination $(1, 2, 5)$ as a parental chromosome to encode and perform mutation in accordance with the required constraint of maintaining the number of MSUs (i.e., three MSUs at 39 ambulance sites), as shown in Fig. 1.

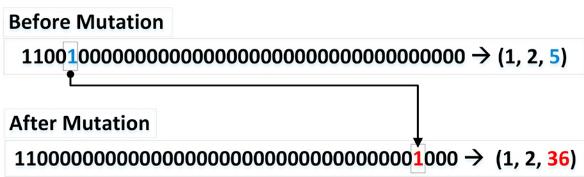


Fig. 1: Mutation operator for the three MSUs setting (old placement in blue and new placement in red).

In the above example, we switch values of two bits (a one-bit and a zero-bit), corresponding to moving one MSU (see before mutation). After applying our designed mutation operation, a new offspring is generated, which involves transferring the flipped bit to a different position (i.e., the 36th bit). Thus, we successfully retained the required number of MSUs.

4.1.3. Crossover

Crossover, also called "recombination", is a genetic operator that combines the genetic information of two chromosomes (often called the "parents") to generate an offspring. In a crossover, we split both chromosomes at a certain point and swapped parts of the corresponding chromosome pieces with another chromosome. Using crossover, we ensure the exchange of genetic characteristics between parents and thus create chromosomes that are more likely to be better than the parents.

In our binary encoding scheme, it is essential to choose a crossover operator that can act in accordance with the given constraints (i.e., controlling the number of 1s, where a ”1” represents an MSU). Conventional crossover operators do not typically consider any constraints; as a result, they may generate offsprings that violate the (desired number of 1’s or MSUs) constraint. To overcome this issue, we found the shuffle crossover a close match to our problem since using this operator allows us to control the number of 1s in a bit string representing MSU placements. Drawing inspiration from the shuffle crossover, we developed a crossover operator based on shuffling the bits between two parental chromosomes using mathematical set operations (i.e., union, intersection, subtraction, and symmetric difference). In this crossover operator, we first randomly shuffle the genes in both parents. Then, we apply the 1-point crossover technique by selecting a midpoint as a crossover point to create two offsprings. To demonstrate our crossover operator, we use the 3MSUs setting as an example. In our example, we select the two MSU locations $(1, 2, 3)$ and $(37, 38, 39)$, which we treat as parental chromosomes. We then perform the crossover operation while ensuring that the required constraint of having three MSUs at 39 ambulance sites is met. The process is illustrated in Fig. 2

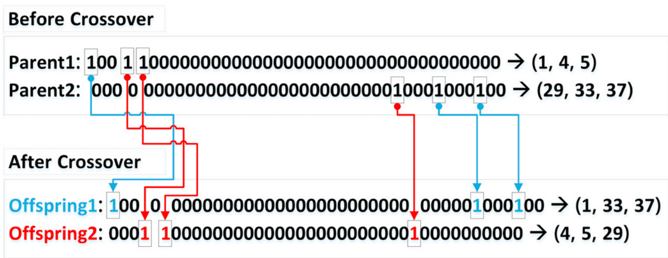


Fig. 2: Crossover operator working for 3MSUs setting (offspring 1 in blue and offspring 2 in red).

In the above example, our designed crossover operator generates offspring solutions by swapping genetic information between two parent solutions. The crossover operator maintains the number of 1s in the generated offspring solutions, ensuring that the number of MSUs in the offspring solutions is the same as that of the parents. This is achieved by selecting a subset of the different bits between the two parent solutions and randomly swapping them between the two offspring solutions while preserving the common bits between the parents. The resulting offspring solutions are then composed of the selected bits from the parents.

4.1.4. Fitness Function

In the GA, the fitness function aims to compute the quality of the chromosomes. To calculate the fitness function, we use the objective function of the optimization model (see Section 3) to evaluate the goodness of the chromosomes (referred to as “MSU locations”) that minimize the expected time to treatment for the entire SHR region (considering both regular ambulances and MSUs). To achieve this, we use the weighted average expected time to treatment, that is,

$$\sum_{r \in R} Q_r \cdot \min\{t_r^{RA}, t_r^{MSU}\}, \tag{5}$$

as our fitness function and the goal is to minimize this function.

4.2. Genetic Algorithm Process

4.2.1. Initialization

We initialized our GA by randomly selecting a set of chromosomes with no bias toward solutions. This helps to ensure diversity in the population since it allows the GA to consider a wide range of solutions, including ones that may not be as fit according to the fitness function but may have characteristics that could lead to better solutions in the future. The number of chromosomes in the initial population is the population size. This step sets the stage for the optimization process and determines the range of solutions the GA will consider.

4.2.2. Selection

After ranking the fitness of the chromosomes in ascending order, we select the top fittest 40% of chromosomes in the population and let them pass to the next generation on the basis of merit "the fitter the chromosome, the higher the survival chance".

4.2.3. Reproduction

Reproduction refers to creating a new generation of solutions from the current population. Using crossover (i.e., combining parts of the parent solutions to create the offspring) and mutation (i.e., introducing small random changes to the offspring), we explore new regions of the search space and potentially discover better solutions to the problem. For reproduction, we select the remaining chromosomes in the population (other than the top 40% fittest) as candidates for crossover and for mutation in a percentage distribution (i.e., 30% for mutation and 30% for a crossover).

4.2.4. Termination

Our termination criteria are based on the number of iterations. We stop the optimization process after creating a certain number of generations, for example, 100 or 200.

5. Computational Study

5.1. Scenario Description

In order to assess the efficiency of our GA optimization method for MSU placements, we applied the method to Sweden's SHR. The SHR covers four counties comprising 49 municipalities, with 13 acute hospitals and 39 ambulance sites. An overview of the SHR is provided in Fig. 3, where the green triangles and purple circles represent the locations of ambulance sites and acute hospitals, respectively.

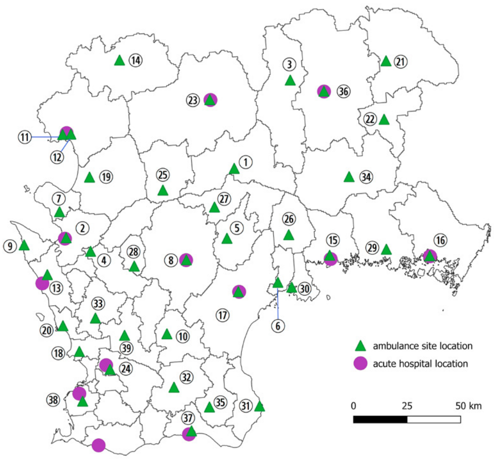


Fig. 3: An overview of the SHR, where ambulance sites and acute hospital locations are shown by green triangles and purple circles, respectively. The circled numbers show the corresponding ambulance site ID.

We used two data types for the analysis: demographic data obtained from Statistics Sweden and stroke-specific data provided by the Southern Regional Health Care committee of Sweden. To facilitate our analysis, we divided the region of interest (ROI) into a set of non-overlapping 1x1 km squares (which is our set R). Please note that the union of all squares $U_{r\epsilon R}$ equals the entire SHR area. The locations of ambulance sites and acute hospitals are acquired using Google Maps and official documentation provided by the healthcare authorities in the region. The stroke data included the number of stroke cases for 21 age groups: $(0, 4), (4, 8), ..., (95, 99), (100, \infty)$. In addition, the demographic data contained the number of inhabitants for each age group and each of the 1×1 *km* squares covering the SHR. The SHR has a population of approximately 1.9 million and an area of 24,000 square kilometers. To show the generalization of our method, we applied our GA method to find the locations of three, four, and five MSUs in the SHR. To represent these settings, we used the notations 3MSUs, 4MSUs, and 5MSUs, respectively. Our method searches for the suitable placement combination and evaluates a different number of potential solutions depending on the number of MSUs. Specifically, we evaluated 9,139 potential solutions in the 3MSUs setting, 82,251 potential solutions in the 4MSUs setting, and 575,757 potential solutions in the 5MSUs setting, using Eq. 4. All of the experimental results were measured in terms of the expected time to treatment.

5.2. Comparison of Genetic Algorithm with Exhaustive Search

To show the effectiveness of our GA method, we compared our results with our companion study by Amouzad Mahdiraji et al. [7], which employs an exhaustive search (ES) to find the optimal sites to place MSUs in the SHR. For comparison analysis, we implemented ES to find optimal fitness function values and reported the results in terms of execution time.

The data presented in Table 1 compares the execution time for two algorithms on 3MSUs, 4MSUs, and 5MSUs settings. Undoubtedly, the GA converges significantly faster than ES in all of the MSUs allocation settings. We also reported the best hyperparameters (population size, mutation rate, and the number of generations), as they have a significant impact on the performance of GA. Careful selection and tuning of these parameters are necessary to increase the possibility of obtaining good results. The GA method dramatically reduces the execution time by 8.75x, 16.36x, and 24.77x when compared to ES in finding optimal placements for 3MSUs, 4MSUs, and 5MSUs, respectively, when compared to ES. For a fair comparison, both algorithms are implemented using Python and tested on an Apple Mac-Book Pro (2021), and the results are compared in terms of execution time (minutes).

Table 1: Execution time of ES and GA for the three, four, and five MSUs settings

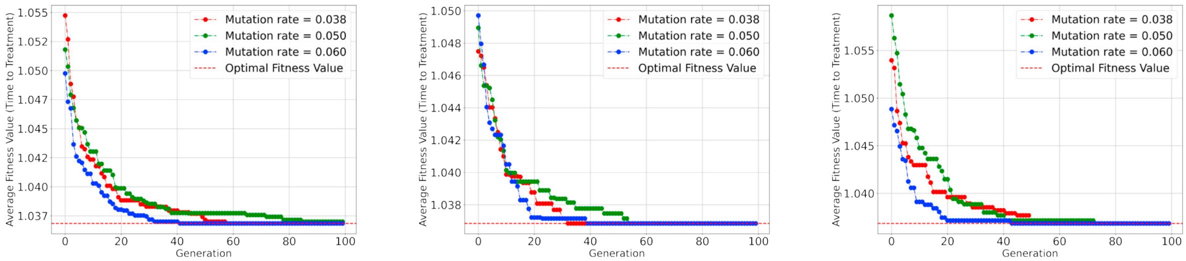
	Exhaustive Search	**Genetic Algorithm**			
			Hyperparamter Settings		
# MSUs	**Execution Time (mins)**	**Execution Time (mins)**	**Population size**	**Mutation Rate**	**# Generations**
3	1.4	0.05	120	0.06	100
4	23.56	1.44	200	0.38	100
5	250.33	10.09	300	0.12	200

5.3. Method Robustness and Sensitivity Analysis

To demonstrate the robustness of our proposed method, we ran the GA 20 times with different random seeds and reported the average results. Additionally, we conducted a sensitivity analysis to highlight the impact of the key hyper-parameters, such as varying mutation rates and population sizes, on the performance and convergence of the GA. To illustrate these results, we considered the 3MSUs setting. Fig. 4 clearly shows that our method converges to the optimal solution for all tested mutation rates and population sizes. These are average performance figures resulting from running our GA method 20 times, which shows that in each run, our method converges to the optimal solution. Moreover, under different hyperparameters, the GA achieves the optimal fitness value of 1.036 across the population sizes 75, 100, and 120 when the mutation rates are 0.038, 0.05, and 0.06, respectively. The fitness values show a general decreasing trend over time, indicating that the algorithm progresses towards the optimal solution. The results of our experiments indicate that the GA is able to consistently find an optimal solution regardless of the mutation rate, as evidenced by achieving the optimal fitness value of 1.036 in all cases. Overall, the 0.06 mutation rate exhibited faster convergence across all population sizes compared to the 0.38 and 0.05 rates. In conclusion, our results indicate that a mutation rate of 0.06 is optimal for the 3MSUs setting.

5.4. Performance Convergence based on the Best Fitness Values - 3MSUs, 4MSUs, and 5MSUs Settings

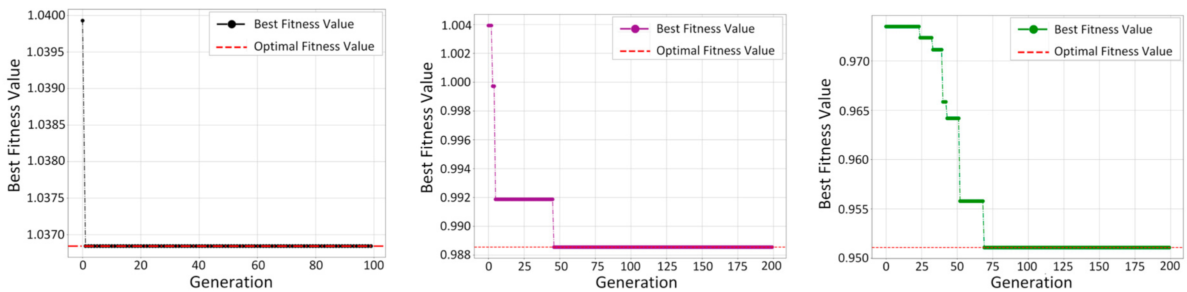
As illustrated in Fig. 5, we conducted experiments for the 3MSUs, 4MSUs, and 5MSUs settings and reported the performance of our method based on the best fitness scores after running the algorithm over multiple generations. The figure clearly shows that in all settings, we efficiently achieved the optimal fitness values – represented by the



(a) Population Size: 75 (b) Population Size: 100 (c) Population Size: 120

Fig. 4: Performance convergence with respect to different mutation rates across multiple population sizes for the 3MSUs setting.

red line on the x-axis in each sub-figure. In Fig. 5(a), the 3MSUs setting, the GA runs for 100 generations, with 20 runs per generation. The best fitness value of each generation was recorded and plotted to visualize the progress of the optimization process. The results of the GA are plotted, with the best fitness value of each generation represented by a black dot-dashed line. We observed that the best fitness value monotonically decreased as the number of generations increased, indicating that the GA was able to find better solutions as it progressed. The best fitness value reached a minimum of 1.036, which is the optimal solution – horizontal red dot-dashed line represents the optimal fitness value. In Fig. 5(b), the 4MSUs setting, the graph shows the evolution of the best fitness value over the generations of the genetic algorithm – x-axis red-dashed line represents an optimal fitness value of 0.988. A similar performance trend is observed for the 5MSUs setting, illustrated in Fig. 5(c) – the x-axis red-dashed line, which represents an optimal fitness value of 0.951.



(a) 3MSUs (b) 4MSUs (c) 5MSUs

Fig. 5: Performance convergence based on the best fitness values across multiple MSUs settings.

6. Conclusion

Our study proposes a novel GA method that efficiently optimizes the placement of MSUs across potential ambulance sites. We demonstrate the effectiveness of our model by significantly outperforming an exhaustive search to solve the MSU allocation problem in terms of execution time. Our experimental results show that the GA method is efficient in finding optimal sites for three, four, and five MSUs – 8.75x, 16.36x, and 24.77x times faster than an exhaustive search. In this method, we designed the complete process of GA according to the MSU allocation problem, which primarily includes an efficient encoding scheme for the input data (the number of MSUs and potential sites) and the design of selection, crossover, and mutation operators in accordance with the optimization problem. We believe our efforts will be helpful in the healthcare domain, particularly in opening the doors for further research on optimal locations for MSUs.

Acknowledgements

This work is financed by the Kamprad Family Foundation for Entrepreneurship, Research & Charity. This work is also partially supported by the Wallenberg AI, Autonomous Systems and Software Program – Humanities and Society (WASP-HS), funded by the Marianne and Marcus Wallenberg Foundation.

References

[1] Comber, A. J., Sasaki, S., Suzuki, H., and Brunsdon, C. (2011). A modified grouping genetic algorithm to select ambulance site locations. International Journal of Geographical Information Science, 25(5), 807-823.

[2] Huang, C. Y., & Wen, T. H. (2014). Optimal installation locations for automated external defibrillators in Taipei 7-Eleven stores: using GIS and a genetic algorithm with a new stirring operator. Computational and mathematical methods in medicine, 2014.

[3] Gupta, H., & Mehrotra, D. (2021, September). Optimizing Ambulance Deployment using Genetic Algorithm. In 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO) (pp. 1-5). IEEE.

[4] Wirsansky, E. (2020). Hands-on genetic algorithms with Python: applying genetic algorithms to solve real-world deep learning and artificial intelligence problems. Packt Publishing Ltd.

[5] Saenger, A. K., & Christenson, R. H. (2010). Stroke biomarkers: progress and challenges for diagnosis, prognosis, differentiation, and treatment. Clinical chemistry, 56(1), 21-33.

[6] Krishnamurthi, R. V., Barker-Collo, S., Parag, V., Parmar, P., Witt, E., Jones, A., ... & Feigin, V. L. (2018). Stroke incidence by major pathological type and ischemic subtypes in the Auckland regional community stroke studies: changes between 2002 and 2011. Stroke, 49(1), 3-10.

[7] Mahdiraji, S. A., Holmgren, J., Mihailescu, R. C., & Petersson, J. (2021). An Optimization Model for the Tradeoff Between Efficiency and Equity for Mobile Stroke Unit Placement. In Innovation in Medicine and Healthcare: Proceedings of 9th KES-InMed 2021 (pp. 183-193). Springer Singapore.

[8] World Stroke Organization (2019) Facts and figures about stroke. https://www.world-stroke.org/world-stroke-day-campaign/why-stroke-matters/learn-about-stroke/ (Accessed May 2023)

[9] The Swedish Stroke Register (2020) Stroke registrations. https://www.riksstroke.org/sve/forskning-statistik-ochverksamhetsutveckling/statistik/registreringar/ (Accessed May 2023)

[10] Zhen, L., Wang, K., Hu, H., & Chang, D. (2014). A simulation optimization framework for ambulance deployment and relocation problems. Computers & Industrial Engineering, 72, 12-23.

[11] Liu, Y., Roshandeh, A. M., Li, Z., Kepaptsoglou, K., Patel, H., & Lu, X. (2014). Heuristic approach for optimizing emergency medical services in road safety within large urban networks. Journal of transportation engineering, 140(9), 04014043.

[12] McCormack, R., & Coates, G. (2015). A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival. European Journal of Operational Research, 247(1), 294-309.

[13] Tlili, T., Abidi, S., & Krichen, S. (2018). A mathematical model for efficient emergency transportation in a disaster situation. The American journal of emergency medicine, 36(9), 1585-1590.

[14] Zaheeruddin, & Gupta, H. (2022). Optimized Ambulance Allocation Using Hybrid PSOGA for Improving the Ambulance Service. IETE Journal of Research, 1-12.

[15] Phan, T. G., Beare, R., Srikanth, V., & Ma, H. (2019). Googling location for operating base of mobile stroke unit in metropolitan Sydney. Frontiers in neurology, 10, 810.

[16] Rhudy, J. P., Alexandrov, A. W., Rike, J., Bryndziar, T., Hossein Zadeh Maleki, A., Swatzell, V., ... & Alexandrov, A. V. (2018). Abstract WP215: Geospatial Visualization of Mobile Stroke Unit Dispatches: A Method to Optimize Service Performance. Stroke, 49(Suppl_1), AWP215-AWP215.

[17] Dahllöf O, Hofwimmer F, Holmgren J, Petersson J (2018) Optimal placement of Mobile Stroke Units considering the perspectives of equality and efficiency. Procedia Computer Science 141:311-318