Artificial Intelligence in Medicine 84 (2018) 23–33

Contents lists available at ScienceDirect

Artificial Intelligence in Medicine

journal homepage: www.elsevier.com/locate/aiim

Chaotic genetic algorithm and Adaboost ensemble metamodeling approach for optimum resource planning in emergency departments

Milad Yousefi [a,*], Moslem Yousefi [b], Ricardo Poley Martins Ferreira [c], Joong Hoon Kim [b], Flavio S. Fogliatto [a]

[a] Industrial & Transportation Engineering Department, Universidade Federal do Rio Grande do Sul – UFRGS, 90035-190, Porto Alegre, RS, Brazil
[b] School of Civil, Environmental and Architectural Engineering, Korea University, Seoul 136-713, Republic of Korea
[c] Departamento de Engenharia Mecânica, Universidade Federal de Minas Gerais – UFMG, Belo Horizonte, Brazil

ARTICLE INFO

Article history:
Received 23 December 2016
Received in revised form 6 September 2017
Accepted 8 October 2017

Keywords:
Simulation-based optimization
Decision support system
Adaboost ensemble metamodel
Chaotic genetic algorithm (GA)

ABSTRACT

Long length of stay and overcrowding in emergency departments (EDs) are two common problems in the healthcare industry. To decrease the average length of stay (ALOS) and tackle overcrowding, numerous resources, including the number of doctors, nurses and receptionists need to be adjusted, while a number of constraints are to be considered at the same time. In this study, an efficient method based on agent-based simulation, machine learning and the genetic algorithm (GA) is presented to determine optimum resource allocation in emergency departments. GA can effectively explore the entire domain of all 19 variables and identify the optimum resource allocation through evolution and mimicking the survival of the fittest concept. A chaotic mutation operator is used in this study to boost GA performance. A model of the system needs to be run several thousand times through the GA evolution process to evaluate each solution, hence the process is computationally expensive. To overcome this drawback, a robust metamodel is initially constructed based on an agent-based system simulation. The simulation exhibits ED performance with various resource allocations and trains the metamodel. The metamodel is created with an ensemble of the adaptive neuro-fuzzy inference system (ANFIS), feedforward neural network (FFNN) and recurrent neural network (RNN) using the adaptive boosting (AdaBoost) ensemble algorithm. The proposed GA-based optimization approach is tested in a public ED, and it is shown to decrease the ALOS in this ED case study by 14%. Additionally, the proposed metamodel shows a 26.6% improvement compared to the average results of ANFIS, FFNN and RNN in terms of mean absolute percentage error (MAPE).

1. Introduction

Congestion in emergency departments (EDs) has been reported for decades as a worldwide problem in the healthcare industry while demand for EDs is increasing [1]. In the U.S. the number of visits to EDs from 1997 increased up to 23% in a decade, reaching about 222 visits per minute in 2007 [2]. EDs are complex systems by nature. They operate 24 h a day, seven days a week, while high operating costs are causes of budget shortages as EDs are obligated to attend to any arriving patient. At the same time, an error in their process may be life threatening.

Resource planning in EDs requires analyzing the whole system and finding the most efficient way to allocate resources. Such analysis is a challenging task because EDs are stochastic environments due to the influence of random variables [3]. For instance, patient arrival can be influenced by weather conditions or air pollution [4]. On the other hand, applying trial and error in ED resource planning may result in irreparable consequences. In order to overcome these issues, decision-makers need to monitor the system at the operational and tactical levels.

Since it is difficult to model EDs analytically, computer simulations are vastly used as a tool to monitor ED behavior [5–7]. Various computer simulations have already been applied to study different aspects of EDs. A study by Gul and Guneri [8] provided a review of simulation applications in normal and disaster conditions in EDs. Computer simulations allow hospital managers to observe system behavior and evaluate alternative system scenarios without interrupting routine operations.

In recent years, combinations of simulation and optimization techniques have been used in abundance to find near-optimum

* Corresponding author.
E-mail address: yousefi.milad@gmail.com (M. Yousefi).

https://doi.org/10.1016/j.artmed.2017.10.002

values of decision variables in complex systems including the healthcare industry [8,9]. Ahmed and Alkhamis [10] presented a discrete simulation combined with optimization to provide a decision-making support system (DSS) in an ED in Kuwait to reduce waiting time and increase patient throughput. They reported a 28% increase in patient throughput and a 40% reduction in waiting time. Cabrera et al. [11] and Taboada et al. [12] presented a DSS using a pure agent-based model and exhaustive search in an ED in Spain. Yeh and Lin [13] presented a simulation model combined with the Genetic Algorithm (GA) to find the near-optimum nursing schedule in an ED in Taiwan. The study reported 43.47% and 43.42% reductions in average queue times.

Selecting a key performance indicator (KPI) in ED simulation-optimization is a controversial subject [14]. There is no rule of thumb to select a KPI and variety can be found in the literature. Although all are related to ED crowding and patient satisfaction, the most commonly used KPIs in this field of study include the number of patients who leave without being seen (LWBS) [15–20], the number of discharged patients [12,20,21,22], length of stay (LOS) [12,15,20–24], time to see a doctor [15,16,23] and average waiting time [15,16,20,23–25].

Although optimization-simulation methods have been successfully applied in different ED-related problems, reviewing the past 10 years' worth of literature shows that a common problem with these methods is that they are computationally expensive to explore the entire search space in real-life optimization problems. In this study, an attempt is made to tackle this problem by applying a robust approximation model to find relationships between the inputs and outputs of the proposed DSS and make the process as fast as possible while maintaining reliable approximation. This type of relationship between inputs and outputs is called a "metamodel" [26].

Metamodels enable researchers to obtain reliable approximate model outputs without running expensive and time-consuming computer simulations. Therefore, the process of model optimization can take less computation time and cost. Simulation-based metamodels enable users to employ these tools in crisis decision-making when a case is serious and there is only a very short response time to manage it [27,28].

Machine learning methods have proven to be efficient in finding the nonlinear relation between the inputs and outputs of simulation models. While machine learning, and in particular neural networks, have been successfully employed for constructing metamodels in recent years, there has not been much effort on systematically increasing metamodel efficiency. The literature has shown that the performance of neural network-based metamodels (or predictors in the case of forecasting problems) can be improved, in some cases drastically, by utilizing an ensemble of different metamodels based on their individual performances [29]. Nevertheless, the potential of ensemble metamodeling has not been explored for the problem at hand. In this study, a robust ensemble algorithm is constructed through a systematic method. Three power machine learning approaches, namely ANFIS, FFNN and RNN are used to build the ensemble metamodel through two well-known ensemble approaches: bootstrap aggregating (bagging) and adaptive boosting (AdaBoost).

In this paper, an agent-based simulation is designed and constructed based on a real case study at an ED in a teaching hospital in the capital of the second largest Brazilian state by population. After verifying and validating the computer simulation model, different metamodels are used to find the metamodel with less error. A GA with a chaotic mutation operator is introduced to decrease the average length of stay (ALOS) in the ED while budget and capacity are considered the problem constraints.

The remainder of this paper is organized as follows: In Section 2, a detailed description of the case study and the problem definition are presented. The proposed metamodel approaches and GA are introduced in Section 3. The results from the simulation-based metamodeling approach, a comparison with other methods and the optimization results are given in Section 4. Finally, Section 5 presents the conclusions and future works.

Table 1
Manchester Triage System (MTS).

Priority level	Color	Safe time until first medical visit
Immediate	Red	Immediately
Very urgent	Orange	Up to 10 min
Urgent	Yellow	Up to 60 min
Standard	Green	Up to 120 min
Non-urgent	Blue	Up to 240 min

2. Risoleta Tolentino Neves emergency department

2.1. System description

Risoleta Tolentino Neves Hospital (RTNH) is a teaching hospital in the capital of Minas Gerais state of Brazil, Belo Horizonte. The ED of RTNH operates 24/7 and receives 162 patients a day on average. The ED contains different sections: pediatrics, orthopedics, suturing, yellow zone and emergency rooms (surgical and clinical). Each of these sections provides services for patients based on their problems. The yellow zone and clinical emergency respectively received the most and least patients among all sections, with 44% and 5% of all patients in the first half of 2016. The main resources in this ED are as follows:

1. Receptionists
2. Triage nurses
3. Doctors
4. Nurses
5. Nurse technicians

At one time, 2 receptionists, 1 triage nurse, 22 doctors, 5 nurses and 29 nurse technicians are working in the ED. Fig. 1 demonstrates the flow of patients in the ED. The procedure starts with patient arrival to the department. Patients may arrive by themselves, by ambulance, or in police custody. All patients except for those in police custody must visit a receptionist to register their personal information. Subsequently, they go to a triage nurse in the triage room. In the triage room, the patient's acuity is checked based on the Manchester Triage System (MTS). MTS categorizes patients into five categories: red, orange, yellow, green and blue [30]. Red is for patients with the highest acuity (most urgent) and Blue is for patients with the lowest acuity (least urgent). Patients in police custody can omit registration and go directly to the triage room. Table 1 shows the levels of patient propriety based on MTS.

Following triage, patients wait for the availability of the section where they need to be treated. Except for the suturing section, all sections contain beds. Patients in any section might go to the laboratory or X-Ray section for further examination and later return to their relevant section. After treatment in each section, the patient may leave the department or, depending on necessity, they go to the observation room. The process in the ED begins with registration for admission and concludes when the patient is released.

2.2. Simulation model

To simulate the mentioned system, an open-source multi-agent modeling environment is used, namely NetLogo 5.3.1 [31]. In order to make a reliable ED simulation model, data collection is needed. The main data collection elements in this simulation are as follows:

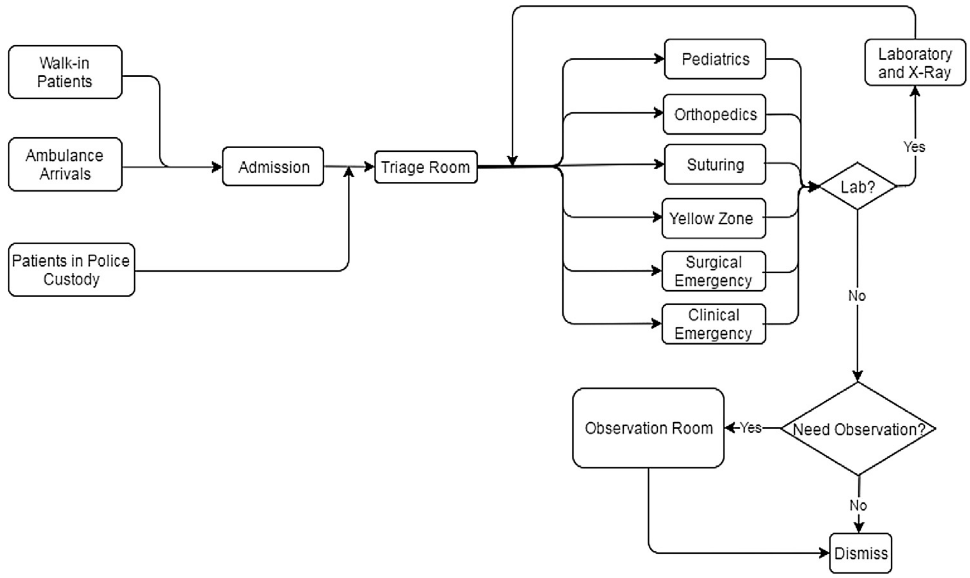

Fig. 1. Flow of patients at the emergency department of the Risoleta Tolentino Neves Hospital.

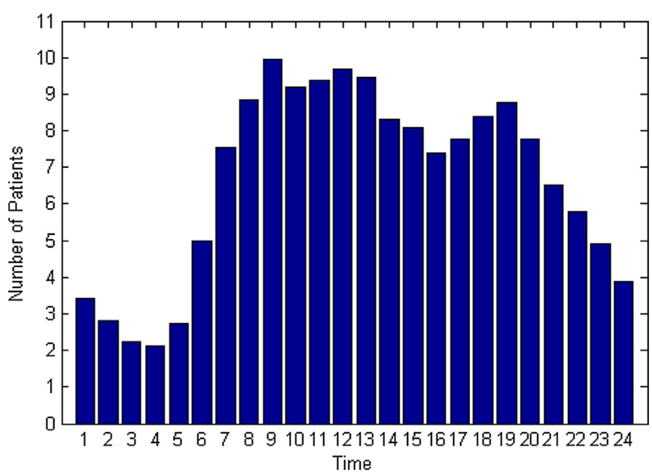

Fig. 2. Patient arrival to the Emergency Department.

the amount of time each service takes (e.g. admission, triage, laboratory, etc.), patient arrival rate and the number of each type of patients in the ED. From January 2016 to May 2016 around 24000 data were collected and analyzed to achieve proper model inputs. One-fourth of the gathered data were used for model validation and 75% served as input to create the model. The patient arrival to the ED is a non-homogenous Poisson process with a rate $\lambda(t)$ and 24 intervals as demonstrated in Fig. 2, where $\lambda(t)$ is the estimated function of patient arrival per hour. Data analysis showed that 44% of patients go to the yellow zone, 30% go to the orthopedics section, 7.8% go to pediatrics and 7% go to surgical emergency, while the suturing section and clinical emergency rates are 6.2% and 5% respectively. Table 2 shows the length of service/treatment in each section, which is estimated by distribution.

To eliminate any bias at the beginning of the simulation, the model was run for 4320 min (3 days) and 100 replications. The first 2880 min (2 days) were set as the warm-up period to make the simulation reach steady state and the remaining time was selected as the study period to collect the results. The confidence interval to compare the simulation results with actual results was set at 95% (a = 0.05). For all performance metrics, the error margin of each confidence interval was calculated by trial and error. To approximate the number of replications, this process continued until the margin of error was less than 5% of the average mean while the simulation computation time was tractable.

Table 2
Service/treatment time in each emergency department section.

No	Section	Distribution (min)
1	Admission	Uniform (3, 6)
2	Triage Room	Uniform (3, 5)
3	Suturing	Triangular (15, 20, 40)
4	Yellow zone	Triangular (10, 20, 25)
5	Orthopedics	Triangular (5, 10, 15)
6	Pediatrics	Triangular (10, 15, 30)
7	Surgical emergency	Triangular (10, 15, 30)
8	Clinical emergency	Triangular (10, 20, 30)
9	Lab and X-Ray	Triangular (15, 30, 45)

2.3. Verification and validation

Verification and validation are two of the most significant steps in any model simulation. Verification entails verifying whether the simulation model works as it should. In fact, verification deals with creating the model correctly. In comparing the animation created in our simulation with the ED routine, the district coordinator of hospital services and the hospital manager verified the current version of the simulation model and confirmed that the output of this simulation can represent a real case study.

Validation concerns building the right model. Therefore, by calibrating the model, comparing its behavior with actual system behavior and repeating this process, the simulation model can be improved until it is acceptable. In this process, a set of data that was not used to create the simulation was used to prove that the simulation model is an accurate representation of a real case study. To validate the simulation model, the total time that patients spent in the ED was extracted from the Emergency Department Information System (EDIS) and compared with the total length of stay from the simulation (Table 3). The comparison validates there were no significant differences between the results obtained using the simulation model for the length of stay of patients in different ED sections and those from the real system (95% confidence level, a = 0.05). Moreover, the simulation throughput for a week was 1112 while the throughput for the real system was 1120. This is also shows the simulation was validated.

Table 3
Comparison of length of stay from simulation and the emergency department information system.

No	Section	Actual time	Simulation time	Confidence interval (95%)
1	Suturing	196.45	202.4	[180.35–212.25]
2	Yellow zone	412.46	401.3	[361.10–429.92]
3	Orthopedics	180.45	192.6	[177.32–199.47]
4	Pediatrics	310.72	318.8	[286.65–342.54]
5	Surgical emergency	438.76	442.6	[401.21–471.52]
6	Clinical emergency	487.41	498.1	[462.12–513.68]

2.4. Resource allocation problem

As mentioned previously, ED resources are extremely restricted. Therefore, the aim of this study is to aid ED managers to allocate their resources in the best way so as to minimize ALOS subject to the problem constraints, which are capacity and budget. The problem can be written as follows:

$$Min\ Z = f\left(X_1, X_2, \ldots, X_n\right), \tag{1}$$

S.t

$$\sum_{j=1}^{i} C_i X_i \le B, \tag{2}$$

$$l_j \le X_j \le u_j \ for\ j = 1, 2, \ldots, 5, \tag{3}$$

$$X_j integer\ for\ j = 1, 2, \ldots, 5. \tag{4}$$

In Eq. (1), Z stands for the total ALOS in the ED and (X_1, X_2, ..., X_n) are decision variables. In fact, Z has no analytical form and it is the output of the simulation model mentioned in previous sections when the decision variables change. The cost of each resource and the budget are denoted by C_i and B respectively. Table 4 illustrates that the capacity levels have an upper bound (u_i) and lower bound (l_i), which are the maximum and minimum capacity levels with a baseline value for the resources in the current ED situation. It should be noted that the same nurses cover three sections, including the yellow zone, orthopedics and pediatrics therefore, later in this paper only one variable is considered.

3. Metamodel-based optimization

3.1. Problem definition

The problem is considered a stochastic simulation-based optimization problem. This type of problems can be expressed mathematically as follows:

$$\min_{x\varepsilon\phi} f\left(x\right) + \varepsilon_{randomness}\left(x\right) \tag{5}$$

s.t.

$$g_i\left(x\right) \le 0 i = 1, 2, \ldots, n \tag{6}$$

In Eq. (5), x and ϕ stand for the design space and a vector of design space, respectively, and the objective function value is $f(x)$. The inherent randomness of simulation output is added to Eq. (5) by adding $\varepsilon_{randomness}$ to the equation. In Eq. (6), $g_i\left(x\right)$ represents the i_{th} constraints of the problem and n is the total number of constraints.

In fact, a metamodel focuses on the relationships between simulation inputs and outputs. The metamodel uses these relationships to approximate the outputs based on the given inputs. In general, the inputs are achieved through the Design of Experiments (DOE) method. The main goal of a metamodel is to make a complex system simpler in reasonable time and with affordable cost. Therefore, metamodel output can be used instead of running a simulation.

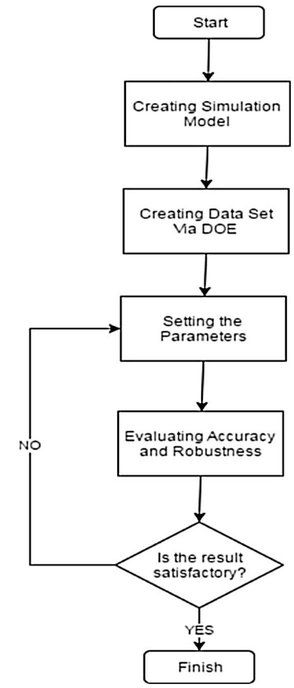

Fig. 3. Process of creating a metamodel.

It should be noted that a metamodel is acceptable only when it does not upset the validity of the simulation. As already mentioned, metamodels provide approximate outputs, meaning that a fitting error should be added to Eq. (5) to make it suitable to represent a metamodel. By adding a fitting error, Eq. (5) should change to Eq. (7) where $\varepsilon_{approximation}\left(x\right)$ is the fitting error.

$$\min_{x\varepsilon\phi} \hat{f}(x) + \varepsilon_{approximation}(x) \tag{7}$$

3.2. The proposed metamodel

The literature indicates that any artificial intelligence approach like ANN or ANFIS has its own advantages. Hence, each can outperform the others in a particular task, something that is well-documented for regression tasks in [32,33]. Therefore in this study, an ensemble of different algorithms is applied to increase metamodel efficiency.

Three well-known machine learning algorithms were constructed and combined through two recognized methods, namely bagging and Adaboost, to determine the best possible metamodel for this task. A conventional feedforward neural network (FFNN), an adaptive neuro-fuzzy inference system (ANFIS) and a recurrent neural network (RNN) were tested. Overall, eight ensemble algorithms were constructed in this study. Fig. 3 demonstrates a schematic of the process of creating a metamodel.

3.2.1. Feedforward neural network (FFNN)

A feedforward neural network contains three principal elements: input, output and hidden layers. Each constitutive unit (artificial neuron) is connected to other neurons in each layer. This connection is mathematically represented by the measure of the connection (weight) between two nodes in the network. In order to minimize the error function, these weights are changed in different steps of the learning process. The learning process eventually leads to the model being able to deal with any unknown sets of data. To select the weights in this artificial neural network (ANN), the Levenberg–Marquardt algorithm (LMA) is implemented. Fig. 4

Table 4
Capacity levels and relative cost of resources in each section.

Section	Resources	Upper bound	Baseline value	Lower bound	Cost (cost unit)
Reception	Receptionist (x_1)	3	2	1	0.45
Triage Room	Triage Nurse (x_2)	3	1	1	0.75
Laboratory and X-Ray	Laboratory Technicians (x_3)	3	3	1	0.6
Suturing	Doctors (x_4)	3	1	1	2.9
	Nurse Technicians (x_5)	3	1	1	0.75
	Nurses (x_6)	1	1	1	1.1
Yellow zone	Doctors (x_7)	5	3	1	2.9
	Nurse Technicians (x_8)	6	4	1	0.75
	Nurses (x_9)	1	1	1	1.1
Orthopedics	Doctors (x_{10})	6	4	1	2.9
	Nurse Technicians (x_{11})	5	3	1	0.75
	Nurses (x_{12})	1	–	1	1.1
Pediatrics	Doctors (x_{13})	4	2	1	2.9
	Nurses Technicians (x_{14})	4	2	1	0.75
	Nurses (x_{15})	1	–	1	1.1
Surgical emergency	Doctors (x_{16})	6	4	1	2.9
	Nurse Technicians (x_{17})	5	3	1	0.75
	Nurses (x_{18})	2	1	1	1.1
Clinical emergency	Doctors (x_{19})	4	2	1	2.9
	Nurse Technicians (x_{20})	7	5	1	0.75
	Nurses (x_{22})	2	1	1	1.1

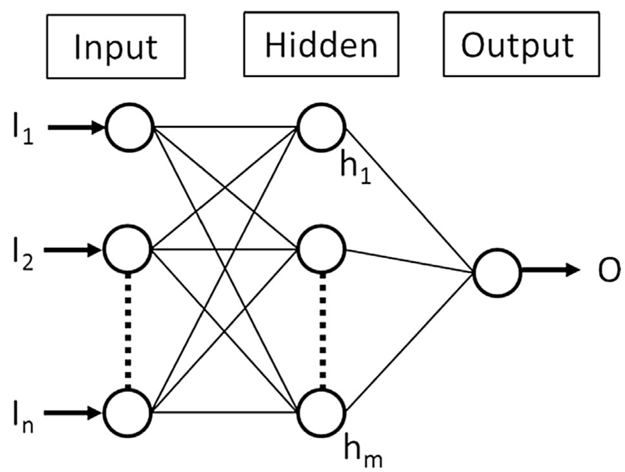

Fig. 4. Diagram of a typical feedforward neural network with one hidden layer.

exhibits different parts of a simple feedforward neural network with one hidden layer.

3.2.2. Recurrent neural network (RNN)

In a conventional ANN, it is assumed that all inputs are independent of each other and hence sequential information is not utilized. Recurrent neural networks (RNNs) are an extension of feedforward networks where each layer has a recurrent connection with a tap delay associated with it. This allows the network to have an infinite dynamic response to time series input data [34]. The term "recurrent" refers to the ability of these networks to do the same task for every element of a sequence. To put it in perspective, RNNs are having a memory to capture and use the information about the previous calculations.

In this study, layer recurrent neural network (LRN) as a variant of RNNs are utilized. Such a recurrent network uses a feedback loop with a single delay. All layers have this feedback loop except for the last layer. LRN is an enhancement of the network previously presented by Elman [35]. In this study, backpropagation, similar to that of FFNN, is used for training the RRN.

3.2.3. Adaptive neuro-fuzzy inference system (ANFIS)

ANFIS architecture benefits from both fuzzy logic and artificial neural networks [36]. The abilities of ANFIS have been proven in different areas of research, including time-series forecasting [37,38], stock market forecasting [39,40] and reducing the bullwhip effect in supply chains [41,42]. In ANFIS, a fuzzy inference system (FIS) is built and a backpropagation algorithm or a combination of a backpropagation algorithm with a least squares method is implemented to adjust the membership function parameters. This adjustment enables ANFIS to learn from the data available in the model.

Similar to neural networks, ANFIS is constructed from a network of input and output layers, where hidden layers connect the input and output layers. These layers are in fact membership functions and other related parameters.

As already mentioned, the process of adjusting the membership functions in ANFIS is the process that causes learning. A gradient vector is used to ease parameter calculation. This gradient vector for any given data set facilitates understanding how the fuzzy inference system maps the input and output data. Actually, it is necessary to apply an optimization method to minimize the error measure by adjusting the parameters once the gradient vector is created. This error measure can be defined as the mean squared difference between expected data and data from a real case. For membership function parameter estimation in ANFIS two approaches can be used: backpropagation or a combination of least squares estimation and backpropagation. Fig. 5 demonstrates a simple five-layer ANFIS with nine if-then rules. This ANFIS has two inputs (x and y) and one output (f).

3.2.4. Ensemble approaches

There are generally two categories of ensemble approaches, namely competitive and cooperative algorithms. In the former, different predictors are used for the task, or the forecast is made based on different data sub-sets and a weighted average of the results is the final forecast. In the cooperative approaches, however, the forecasting task is divided into different sub-tasks where each sub-task is forecasted individually. The final result is the sum of all outputs [43]. An example of a cooperative approach for times series forecasting is wavelet decomposition to pre-process the data set and then employing a predictor to forecast the resultant data sets. The predicted data sets are then aggregated to achieve the forecast values [44].

Among various competitive approaches, bootstrap aggregating (bagging) and adaptive boosting (AdaBoost) have been shown to greatly affect the performance of machine learning approaches for

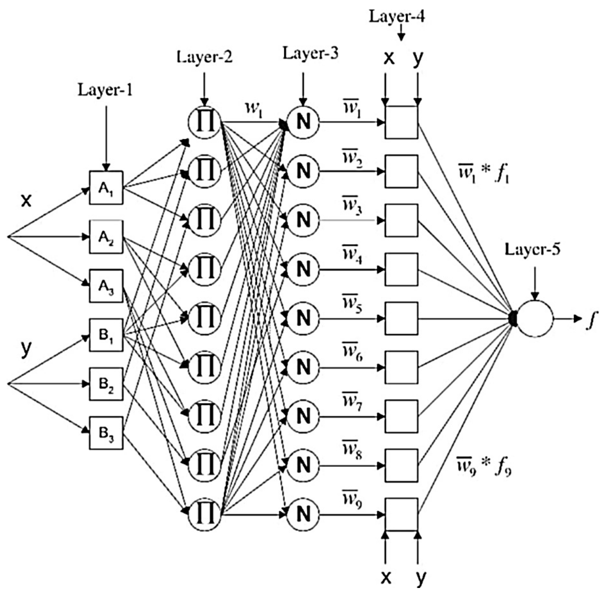

Fig. 5. Flowchart of five-layer ANFIS with two inputs and one output.

both classification and regression tasks. Therefore, these two methods were selected to build a powerful ensemble metamodel.

3.2.4.1. Bootstrap aggregating (bagging). Conventionally abbreviated as bagging, bootstrap aggregating basically entails training many models on separate training sets, hence providing data diversity, and then summing their results with the same weights to achieve a better model [45,46]. Bagging can prevent learning algorithm overfitting as random sampling is performed to train the initial algorithms. In this study, an N model is trained on N randomly drawn data sets with similar sizes. The N sub-data sets are drawn using the sampling with replacement method. Each individual metamodel is assigned a (1/N) weight and the bagging metamodel is built as follows:

$$F_T(x) = \left(\frac{1}{m}\right)\sum_{t=1}^{m} f_t(x) \tag{8}$$

Where m is the number of approaches used in the ensemble method and F_T is the outcome of a given approach.

3.2.4.2. Adaptive boosting (AdaBoost). AdaBoost is an adaptive ensemble approach specifically proposed for classification problems by Freund and Schapire [47]. AdaBoost has mostly been used for classification problems, but recently, its successful application for regression and forecasting tasks has also been reported in literature [48]. Similar to bagging, a number of machine learning algorithms, which are called “weak learners,” are combined into a weighted sum to achieve better results than the weak learners. AdaBoost is different from bagging firstly because the sub algorithms do not have an equal weight in the final ensemble. Secondly, Adaboost is adaptive as it capitalizes on the samples with bigger errors and better-performing weak learners. Adaboost gives higher weightage to samples with poor training results and learners with good performance, and lower weightage to samples with good training results and learners with poor learning performance. Therefore, two sets of weights are to be adjusted in building an AdaBoost ensemble, namely the weights of the training samples and of the weak learners. The final predictor is built as follows:

$$F_T(x) = \sum_{t=1}^{m} \alpha_t f_t(x) \tag{9}$$

Where x is the input of the metamodel, F_T is the final ensemble, t is the number of weak learners, α_t is the weight of each weak learner and f_t is a weak learner.

In the beginning, the weak learners are trained on the same training set. Then the weighted ensemble is created by adding the weak learners to the ensemble one by one. The weight of all samples is assumed to be the same in the beginning and then based on this weight the samples are adjusted throughout the ensemble process.

Assuming $X=[x_1,\ldots x_n]$ represents the training samples, $Y=[y_1,\ldots y_n]$ is the desired output and $w=[w_{1,1},\ldots, w_{n,1}]=1/n$ denotes the initial sample weights, the AdaBoost algorithm used in this study is summarized as follows.

When building an ensemble of m algorithms, in step m-1 the weighted sum of the algorithms is:

$$F_T(x_i) = \alpha_1 f_1(x_i) + \alpha_2 f_2(x_i) + \ldots + \alpha_{m-1} f_{m-1}(x_i) \tag{10}$$

Then for the m-th weak learner, the ensemble is extended to include the weak learner f_m such that by adjusting its corresponding weight, α_m, the total metamodel error is reduced. The AdaBoost computational steps are given as follows.

Step 1: Individually train the weak learners on the data set.

Step 2: Use the build learner f_t to predict the values of data set X_i and the associated modeling error for each sample as follows:

$$\xi_i = \frac{|y_i - \hat{y}_i|}{y_i} \tag{11}$$

$$\xi_m = \frac{1}{n}\sum_{i=1}^{n} \xi_i \tag{12}$$

Step 3: Calculate the weight of weak learner f_m:

$$\alpha_m = \frac{1}{2}\ln\left(\frac{1-\xi_m}{\xi_m}\right) \tag{13}$$

Step 4: Update the sample weights as follows:

$$w_m(i) = \frac{w_{m-1}(i)\,\beta_m^{-\xi_i}}{Z_m} \tag{14}$$

$$\beta_m = \frac{\xi_m}{1-\xi_m} \tag{15}$$

Where Z_m is a normalizing factor so the sum of all weights throughout the sample becomes 1.

Step 5: Repeat steps 2–4 until all weak learners are added to the ensemble algorithm.

Four ensemble algorithms were tested in this study: FFNN-RNN (FR), FFNN-ANFIS (FA), RNN-ANFIS (RA) and FFNN-RNN-ANFIS (FRA). For each set of algorithms, the ensembles were created with both bagging and AdaBoost, therefore a total of eight ensemble algorithms were tested to find the best metamodel for this problem.

3.2.5. Network performance evaluation

Metamodel performance was evaluated using two statistical indicators, namely mean absolute percentage error (MAPE) and coefficient of determination (R^2). MAPE is calculated as follows:

$$MAPE = \frac{\sum_{i=1}^{n}\left(\left|\frac{M_i - E_i}{M_i}\right| \times 100\right)}{n} \tag{16}$$

Where n is the number of observations, E_i is the metamodel result, and M_i is the desired simulation result. MAPE provides information on metamodel performance, with lower values indicating better performance.

The coefficient of determination (R^2) is simply the square of the sample correlation coefficient between the metamodel outcome and the desired simulation result. This coefficient varies in

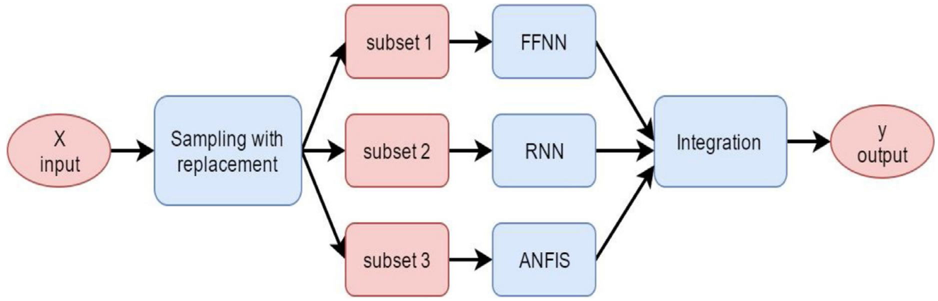

Fig. 6. Flowchart of the three-algorithm bagging ensemble approach.

the range of 0–1. The higher the value, the better the metamodel performance is (Fig. 6).

3.3. Design of experiments

The most accurate way to study the behavior of a simulation model and find the best configuration of resources may be to use complete enumeration. To do so, it would be necessary to perform a large number of experiments to cover all parts of the space in the domain of decision variables. Therefore, design of experiments (DOE) is used to find a sample that covers most space parts with the least number of simulation executions. In fact, DOE can be defined as a matrix, where columns represent factors (variables) and rows represent a sample [49].

There are different DOE and sampling methods, the most common being the 2^k factorial design where k is number of factors [50]. For instance, as our simulation has 19 variables the number of experiments is $2^{19} = 524288$. Suppose we need 50 replications for each experiment and each run takes 60 s, we would need almost 50-year CPU time. When the number of factors increases, the model proposed by McKay et al. [51] and Iman and Conover [52] is more efficient. Latin hypercube design (LHD) is a matrix with m rows and k columns, where m is the number f design points and k is number of variables. A rule of thumb for choosing the number of design points is m = 10 k. Hence, in our case $10 \times 19 = 190$ experiments are needed. For more details regarding LHD see [53,54].

Generally, more samples in metamodel training ensure better system performance but our network needs to be over-trained. To cope with this issue, LHD was employed in this study to create 3 different samples including 250, 500, 750 and 1000 experiments called M_1, M_2, M_3 and M_4 respectively. Subsequently, simulations of each of these design points were run to achieve the study objective (average total time).

3.4. Chaotic genetic algorithm

After creating the simulation model and making the metamodel, the next step is to apply an optimization approach to minimize the ALOS of patients subject to budget and capacity constraints. Fig. 7 illustrates the optimization approach proposed in this paper.

Genetic algorithms (GAs) [55] mimic the process of natural selection, and similar to any other population-based algorithm, begin with generating a group of possible solutions. Each possible solution is called a chromosome in GA. The fittest chromosomes create the next generation with a crossover operator that makes them give part of their genes to the next generation, while a mutation operator helps keep the diversity of possible solutions high enough to avoid premature convergence.

Fig. 8 shows the chromosomes used in this study. Each column, dubbed as a gene, consists of a number corresponding to a specific resource in the ED. Each chromosome must meet the pre-defined capacity and budget constraints. Any chromosome with a budget equal to or less than 68 cost units is feasible, provided that all its genes are in the prescribed range as described in Table 3. To evaluate the fitness of a chromosome a fitness function is used, which is replaced with a metamodel in this study.

To avoid trapping into local optima, a chaotic mutation operator is implemented in the GA similar to [56]. This mutation operator is easy to implement and computationally inexpensive. A one-dimensional logistic map is implemented for the mutation operator to generate a new solution. All elements of a chromosome are first scaled to the range of 0–1 and then a new chromosome is iteratively generated by using the classic logistic chaotic sequence as follows:

$$X_i^{(n+1)} = 4X_i^{(n)}\left(1 - X_i^{(n)}\right) \tag{17}$$

$$0 \le X_i^{(n)} \le 1 \tag{18}$$

$$X_0^{(n)} \ne 0, 0.25, 0.75 \, and \, 1 \tag{19}$$

Where n is the iteration number and k denotes the total number of chaotic variables. Having determined the new sequence, the variables are mapped to their respective real values.

Additionally, a penalty function is added to the fitness function to handle the constraints as GAs are generally unable to do so. A simple penalty function approach is implemented in this study. Selecting a high penalty parameter would eliminate the infeasible solutions from the search space, while a low penalty may result in infeasible final solutions. Therefore, the penalty parameters are set through an extensive trial and error process.

After several algorithm executions and observing the solution evaluation, the initial population was set to 100 while the mutation and crossover rates were set to 10% and 75% respectively. The results show that the chance of improvement after 80 generations is low; therefore, 90 generations were selected as the termination criterion.

4. Computational results

4.1. Data set size

Before creating the ensemble algorithms, the size of the data set to be used for the metamodel should be selected. A low number of training data could result in low metamodel accuracy, while an excessive number of samples may yield overfitting. Moreover, computation time increases with sampling size. A primary metamodel was created with four training sample sizes, namely $M_1 = 250$, $M_2 = 500$, $M_3 = 750$ and $M_4 = 1000$. For all cases, the test data set size was 200 and the samples were independent of those used for training.

4.2. FFNN

A three-layer feedforward neural network consisting of one input, one output and one hidden layer was utilized in this study.

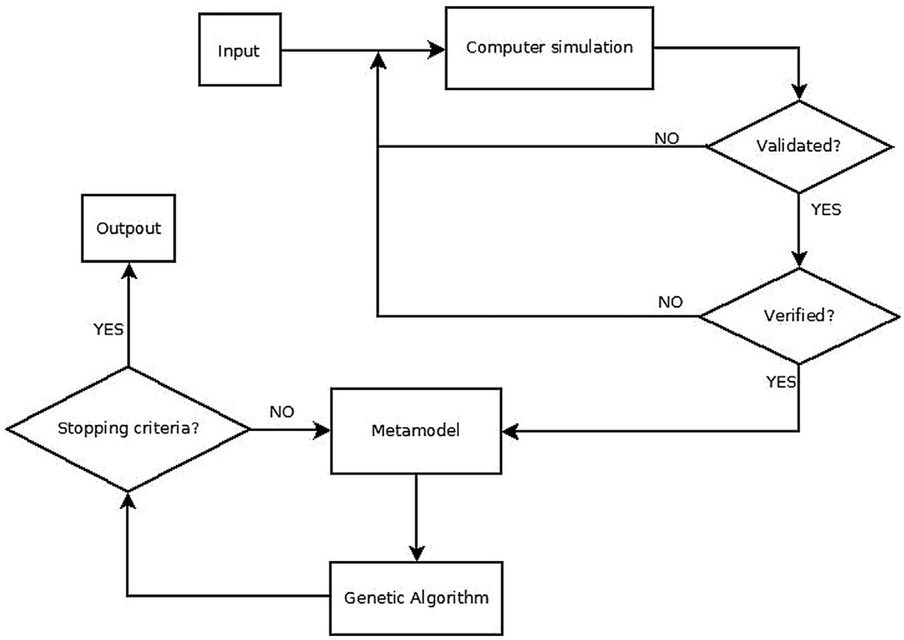

Fig. 7. Combined optimization-simulation approach using the metamodel.

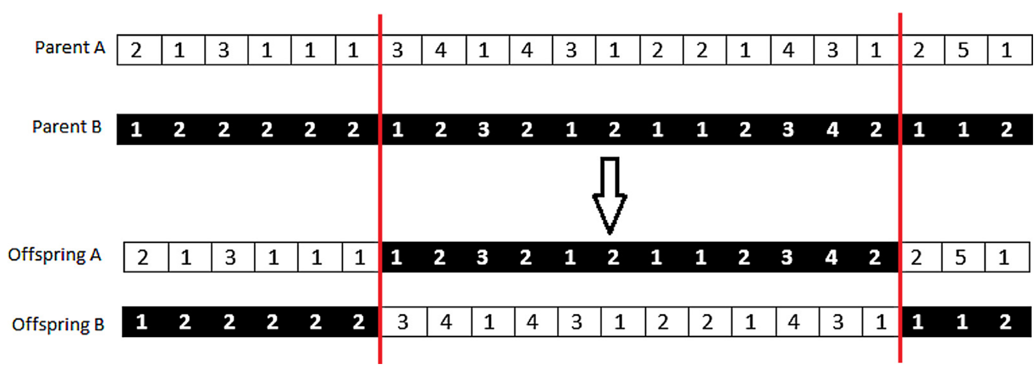

Fig. 8. Genetic algorithm crossover process.

The input is a set of vectors that represent the decision variables of the simulation model and the output layer is a single point representing the simulation output. The activation functions used for the hidden and output layers are the hyperbolic tangent (tansig) and linear (pure-lin) functions, respectively. Three training algorithms, i.e. Levenberg–Marquardt (LM), Resilient Backpropagation (RB) and Scaled Conjugate Gradient (SCG) were initially tested and owing to the better performance of the LM algorithm, it was chosen for the metamodel. The number of neurons in the hidden layer was selected by trial and error, as is the case in literature. Through extensive experimentation, the optimum neural structures were found to be 5, 7, 8 and 10 for cases M_1, M_2, M_3 and M_4 respectively.

4.3. RNN

Similar to FFNN, Levenberg–Marquardt served as the training algorithm and the network structure was found through trial and error. The RNN network was created in MATLAB with an available customized function. The input and output data set is similar to that of FFNN (similar size) and the optimum number of neurons were found to be 5, 7, 8 and 10 for cases M_1, M_2, M_3 and M_4 respectively.

4.4. ANFIS

To develop an ANFIS for the metamodel task, which is similar to a prediction problem, the same approach as in [57] was employed. An initial fuzzy model was required to find the number of inputs, number of linguistic variables and consequently, the number of fuzzy rules in the final model.

The subtractive clustering method proposed by Chiu [58] was applied to the input–output data pairs to extract the initial fuzzy model. It is a fast unsupervised method for estimating the number of clusters and their cluster centers by measuring the potential of data points in the feature space. Having estimated the clusters, the number of fuzzy rules and the premise fuzzy membership functions should be determined. For models M_1, M_2, M_3 and M_4, a Gaussian membership function with 4, 5 and 9 and 12 fuzzy rules, respectively, was considered. The number of fuzzy rules was attained through an optimization procedure using the least squares method and a hybrid learning algorithm.

The MAPE and R^2 of the three algorithms with four data sets with sizes of 250, 500, 750 and 1000 were determined. The results presented in Table 5 show that in the training phase, all algorithms performed better, considering both MAPE and R^2 indices,

Table 5
Performance of FFNN, RNN and ANFIS with different training data sets.

	R^2				MAPE (%)			
	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
Training								
FFNN	0.9836	0.9839	0.9889	0.9907	3.6416	3.4402	3.4378	3.227
RNN	0.9764	0.9771	0.9785	0.9903	4.3241	4.1287	4.0952	3.4791
ANFIS	0.9771	0.9825	0.9892	0.9895	3.5712	3.3601	3.3319	3.3041
Test								
FFNN	0.9752	0.9816	0.9852	0.9641	3.8924	3.6601	3.6584	4.2422
RNN	0.9690	0.9710	0.9787	0.96004	4.9567	4.3475	4.1002	4.3674
ANFIS	0.9753	0.9812	0.9875	0.9883	3.8736	3.5733	3.5005	3.5891

Table 6
Performance of ensemble algorithms on training and test data.

Ensemble		R^2		MAPE (%)	
		Training	Testing	Training	Testing
FR	Bagging	0.9912	0.9896	3.2678	3.4981
	AdaBoost	0.9915	0.9901	3.002	3.1782
FA	Bagging	0.9919	0.9899	3.2532	3.4467
	AdaBoost	0.9927	0.9903	2.9802	3.131
RA	Bagging	0.9914	0.9890	3.2789	3.4490
	AdaBoost	0.9917	0.9895	2.9887	3.1672
FRA	Bagging	0.9923	0.9900	3.2541	3.4056
	AdaBoost	0.9943	0.9913	2.9185	2.9865

with increasing input data set size. However, the trend is different in the test data set where all algorithms presented the least MAPE error on M_3 data set and all but ANFIS had a higher R^2 on M_3 data set. These results indicate that all algorithms suffered from overfitting when the size of the data set had increased from 750 to 1000 instances. Therefore, the M_3 data set with a size of 750 samples was selected for building the ensemble algorithms. The best training and testing performances of all ensemble algorithms are shown in bold font in Table 6.

For the bagging ensemble, the training data set with 2000 samples was used to randomly create three 750-sample data sets using the sampling method with replacement.

For AdaBoost, one 750-sample data set was randomly chosen from the available training samples. The ensemble approach results are presented in Table 6. It is observed from these results that for all ensemble algorithms, AdaBoost outperformed bagging in both training and testing phases. The least MAPE error was achieved for the AdaBoost ensemble of the three algorithms, i.e. FFNN, RNN and ANFIS. Moreover, the AdaBoost ensemble had the highest coefficient of determination (R^2).

4.5. Optimization

As mentioned before, a chaotic mutation operator was implemented to tackle this problem. Studying the system behavior signifies that the yellow zone section has the most effect on ALOS because not only did it receive the majority of patients but the treatment in the yellow zone was also time consuming. As illustrated in Table 7, the yellow zone resources increased more than other sections. Fig. 9 shows the impact of changing the number of doctors and nurse technicians in the yellow zone while other resources remain the same.

Another important factor in improving emergency department performance is the number of triage nurses (Fig. 10).

The Adaboost ensemble of FFNN/RNN/ANFIS (dubbed as Adaboost FRA) is the most efficient metamodel for our problem. It was thus selected to function as the objective function of GA to find the best ED resource planning considering a 68cost unit in order to reduce the patient ALOS in the ED. This tactical decision-making

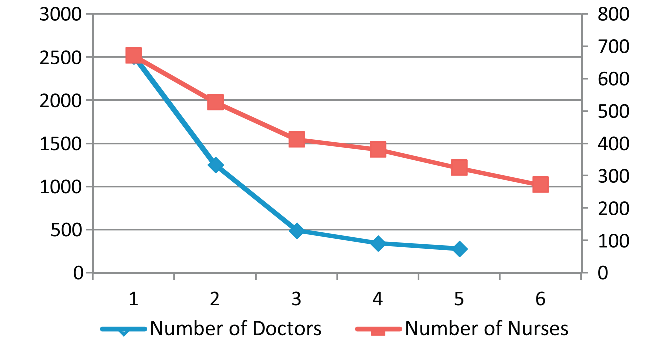

Fig. 9. Impact of changing the number of doctors and technician nurses in the yellow zone on the total length of stay. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

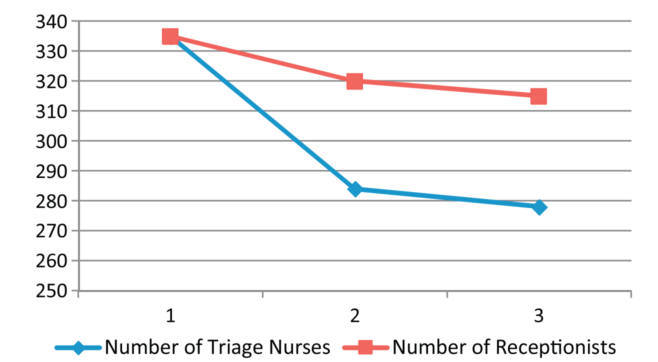

Fig. 10. Impact of changing the number of triage nurses on the average length of stay.

process is very important for hospital decision-makers because the more time patients spend in the ED, the more resources are needed. Furthermore, a long LOS can lead to patient dissatisfaction. Table 7 shows a comparison of the current ED resource allocation with the results obtained from optimization approach.

The results from the current ED resource planning and near-optimum resource allocation from optimization approach are compared in Fig. 11. The patient ALOS with the new resource allocation shows a 14% reduction on average.

5. Concluding remarks

This study was intended to present a robust and computationally practical simulation-based optimization tool for resource planning in emergency departments. Agent-based simulation and an ensemble metamodel were used to provide a decision support system in an emergency department. A genetic algorithm (GA) was then implemented to find the near-optimum resource allocation for emergency departments.

This paper provided a framework to efficiently combine simulation and metamodels in the healthcare industry, because combining these two tools can enable the management to make real-time decisions based on accurate simulation results in a reasonable time.

A robust ensemble of various machine learning algorithms was constructed through a systematic method by using two well-known ensemble approaches, namely bagging and Adaboost. A conventional feedforward neural network (FFNN), adaptive neuro-fuzzy inference system (ANFIS) and recurrent neural network (RNN) were used for this task. Overall, eight different ensembles of these algorithms were constructed and the results indicated that an Adaboost ensemble of ANFIS/FFNN/RNN performs the best. This ensemble metamodel showed a 26.6% improvement compared to

Table 7
Comparison between baseline model and near-optimum resource planning after optimization.

Sections	Resources	Number of each resource in baseline scenario	Number of each resource after optimization
Reception	Receptionists (x_1)	2	2
Triage Room	Triage nurses (x_2)	1	2
Lab and X-ray	Laboratory technicians (x_3)	3	2
Suturing	Doctors (x_4)	1	1
	Nurse technicians (x_5)	1	1
	Nurse (x_6)	1	1
Yellow Zone	Doctors (x_7)	3	5
	Nurse technicians (x_8)	4	6
	Nurses (x_9)	1	1
Orthopedics	Doctors (x_{10})	4	2
	Nurses technicians (x_{11})	3	3
Pediatrics	Doctors (x_{12})	2	2
	Nurses (x_{13})	2	2
Surgical Emergency	Doctors (x_{14})	4	3
	Nurses technicians (x_{15})	3	3
	Nurses (x_{16})	1	1
Clinical Emergency	Doctors (x_{17})	2	3
	Nurse technicians (x_{18})	5	3
	Nurses (x_{19})	1	1

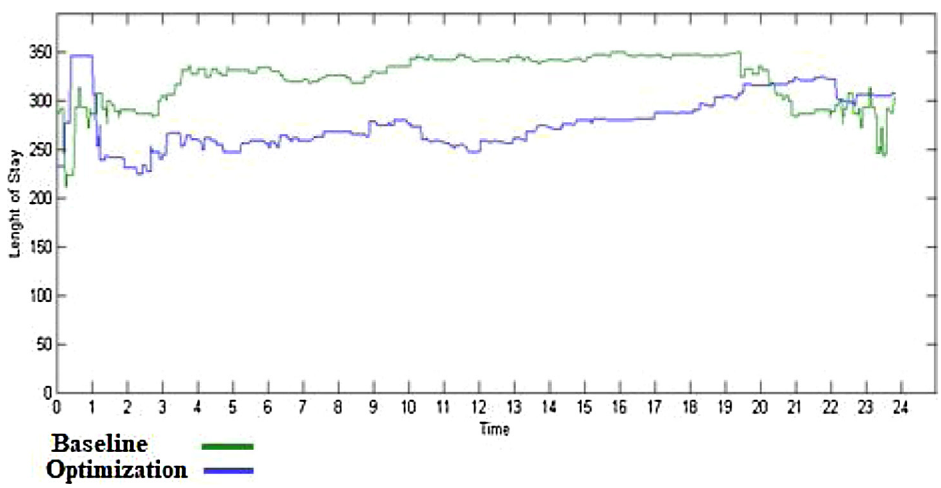

Fig. 11. Comparison of length of stay in baseline resource planning and the genetic algorithm results.

the average results of ANFIS, FFNN and RNN in terms of the MAPE indicator.

Consequently, this powerful ensemble metamodel served as the fitness function of the proposed GA to identify optimum ED resource allocation. Since the optimization involved various variables, nineteen in this case study, a powerful optimization algorithm was needed for the task. Though successfully employed in many tasks, GAs have a tendency of getting trapped in local optima. Hence, in this study, a chaotic mutation operator was added to the GA. This mutation operator is simple to apply and does not add much computational burden. The optimization algorithm was used to decrease the ALOS by adjusting twenty-one variables, including resources, e.g. number of doctors, nurses, receptionists, etc.

A simulation model of the Emergency Department at Risoleta Tolentino Neves Hospital in the capital of Minas Gerais state of Brazil, Belo Horizonte, was designed and constructed. A simulation result analysis signified this ED has some bottlenecks, including the Yellow zone and triage room. The proposed GA-based optimization approach was thus tested in this public ED and was shown to decrease the ALOS in this ED case study from 5.47 h to 4.75 h.

This study highlighted ALOS, but the impact of the same approach on other KPIs including waiting time, number of patients who leave the ED without being seen, etc., could be considered in future studies. Future studies may also implement new approaches for the design of experiments and different optimization algorithms for the same problem. Furthermore, we designed and implemented an agent-based simulation in an ED without considering many agent characteristics. Extending this agent-based simulation for ED modeling while patients can transmit illnesses to other patients is another possible area of research for future studies.

Acknowledgements

The authors would like to thank Dr. Lucas Coimbra, Coordinator of the Risoleta Tolentino Neves Hospital Emergency Department for assistance with the simulation design and data collection, and Dr. Henrique Torres, General Manager of the Hospital for comments that greatly improved the manuscript.

References

[1] Sørup CM, Forberg JL. Development of a Generic Performance Measurement Model in an Emergency Department. Doctoral dissertation, Department of Management Engineering, Technical University of Denmark); 2015.
[2] Niska R, Bhuiya F, Xu J. National hospital ambulatory medical care survey: 2007 emergency department summary. Natl Health Stat Rep 2010;26(26):1–31.
[3] Bagust A, Place M, Posnett JW. Dynamics of bed use in accommodating emergency admissions: stochastic simulation model. BMJ 1999;319(7203):155–8.
[4] Holleman DR, Bowling RL, Gathy C. Predicting daily visits to a waik-in clinic and emergency department using calendar and weather data. J Gen Intern Med 1996;11(4):237–9.

[5] Amaral TM, Costa AP. Improving decision-making and management of hospital resources: an application of the PROMETHEE II method in an emergency department. Oper Res Health Care 2014;3(1):1–6.
[6] Kuo YH, Rado O, Lupia B, Leung JM, Graham CA. Improving the efficiency of a hospital emergency department: a simulation study with indirectly imputed service-time distributions. Flexible Serv Manuf J 2016;28(1–2):120–47.
[7] Ahalt V, Argon NT, Ziya S, Strickler J, Mehrotra A. Comparison of emergency department crowding scores: a discrete-event simulation approach. Health Care Manage Sci 2016:1–12.
[8] Gul M, Guneri AF. A comprehensive review of emergency department simulation applications for normal and disaster conditions. Comput Ind Eng 2015;83:327–44.
[9] Ghanes K, Wargon M, Jouini O, Jemai Z, Diakogiannis A, Hellmann R, et al. Simulation-based optimization of staffing levels in an emergency department. Simulation 2015:942–53.
[10] Ahmed MA, Alkhamis TM. Simulation optimization for an emergency department healthcare unit in Kuwait. Eur J Oper Res 2009;198(3):936–42.
[11] Cabrera E, Taboada M, Iglesias ML, Epelde F, Luque E. Optimization of healthcare emergency departments by agent-based simulation. Procedia Comput Sci 2011;4:1880–9.
[12] Taboada M, Cabrera E, Iglesias ML, Epelde F, Luque E. An agent-based decision support system for hospitals emergency departments. Procedia Comput Sci 2011;4:1870–9.
[13] Yeh J-Y, Lin W-S. Using simulation technique and genetic algorithm to improve the quality care of a hospital emergency department. Expert Syst Appl 2007;32(4):1073–83.
[14] Ghanes K, Diakogiannis A, Jouini O, Jemai Z, Wargon M. Key performance indicators for emergency departments: a survey from an operations management perspective. In: Working paper. Paris: Ecole Centrale; 2014.
[15] Miller M, Ferrin D, Shahi N. Estimating patient surge impact on boarding time in several regional emergency departments. In: Proceedings of the 2009 winter simulation conference (WSC). 2009.
[16] Medeiros DJ, Swenson E, DeFlitch C. Improving patient flow in a hospital emergency department. In: Proceedings of the 40th conference on winter simulation. 2008.
[17] Ferrin DM, Miller MJ, McBroom DL. Maximizing hospital finanacial impact and emergency department throughput with simulation. In: Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come. 2007.
[18] Powell E, Khare R, Reinhardt G. 223: using computer simulation to evaluate the effect of point-of-care testing on emergency department patient flow. Ann Emerg Med 2007;50(3):S70.
[19] Bair AE, Song WT, Chen YC, Morris BA. The impact of inpatient boarding on ED efficiency: a discrete-event simulation study. J Med Syst 2010;34(5):919–29.
[20] Yousefi M, Ferreira RPM. An agent-based simulation combined with group decision-making technique for improving the performance of an emergency department. Braz J Med Biol Res 2017;50(5).
[21] Centeno AP, Martin R, Sweeney R. REDSim: a spatial agent-based simulation for studying emergency departments. In: 2013 winter simulations conference (WSC). 2013.
[22] Chonde S, Parra C, Chang C-J. Minimizing flow-time and time-to-first-treatment in an emergency department through simulation. In: Proceedings of the 2013 winter simulation conference: simulation: making decisions in a complex world. 2013.
[23] Sengupta S, Deneweth M, Van Til RP. A better approach to modeling emergency care service. In: Proceedings of the winter simulation conference. 2011.
[24] Abo-Hamad W, Arisha A. Simulation-based framework to improve patient experience in an emergency department. Eur J Oper Res 2013;224(1):154–66.
[25] Hoot NR, LeBlanc LJ, Jones I, Levin SR, Zhou C, Gadd CS, et al. Forecasting emergency department crowding: a discrete event simulation. Ann Emerg Med 2008;52(2):116–25.
[26] Kleijnen JP. A comment on Blanning's metamodel for sensitivity analysis: the regression metamodel in simulation. Interfaces 1975;5(3):21–3.
[27] Kleijnen JP, Sargent RG. A methodology for fitting and validating metamodels in simulation. Eur J Oper Res 2000;120(1):14–29.
[28] Barton RR, Meckesheimer M. Metamodel-based simulation optimization. Handbooks in operations research and management science, vol. 13; 2006. p. 535–74.
[29] Hansen LK, Salamon P. Neural network ensembles. IEEE Trans Pattern Anal Mach Intell 1990;12:993–1001.
[30] Mackway-Jones Kevin, Jill Windle JM. Emergency triage manchester triage group. 3rd ed. United Kingdom: John Wiley & Sons, Ltd.; 2014. p. 204.
[31] Wilensky U, Evanston I. NetLogo. Center for connected learning and computer-based modeling. IL: Northwestern University Evanston; 1999. p. 49–52.
[32] Taskaya-Temizel T, Casey MC. A comparative study of autoregressive neural network hybrids. Neural Netw 2005;18(5):781–9.
[33] Damousis P, Dokopoulos IGP. A fuzzy expert system for the forecasting of wind speed and power generation in wind farms. In: Innovative computing for power-electric energy meets the market. 22nd IEEE power engineering society international conference on PICA Power Industry Computer Applications. 2001. p. 2001.
[34] Mikolov T, Karafiát M, Burget L, Cernocký J, Khudanpur S. Recurrent neural network based language model. Paper presented at the interspeech 2010.
[35] Elman JL. Finding structure in time. Cognit Sci 1990;14(2):179–211.
[36] Jang J-S. ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 1993;23(3):665–85.
[37] Wang WC, Chau KW, Cheng CT, Qiu L. A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series. J Hydrol 2009;374(3):294–306.
[38] Firat M, Güngör M. Hydrological time-series modelling using an adaptive neuro-fuzzy inference system. Hydrol Process 2008;22(13):2122–32.
[39] Boyacioglu MA, Avci D. An adaptive network-based fuzzy inference system (ANFIS) for the prediction of stock market return: the case of the Istanbul stock exchange. Expert Syst Appl 2010;37(12):7908–12.
[40] Kazem A, Sharifi E, Hussain FK, Saberi M, Hussain OK. Support vector regression with chaos-based firefly algorithm for stock market price forecasting. Appl Soft Comput 2013;13(2):947–58.
[41] Tozan H, Vayvay O. Hybrid grey and ANFIS approach to bullwhip effect in supply chain networks. WSEAS Trans Syst 2009;8(4):461–70.
[42] Zarandi MF, Gamasaee R. A type-2 fuzzy system model for reducing bullwhip effects in supply chains and its application in steel manufacturing. Sci Iranica 2013;20(3):879–99.
[43] Dietterich TG. Ensemble methods in machine learning. In: International workshop on multiple classifier systems. Springer; 2000.
[44] Ren Y, Suganthan P, Srikanth N. Ensemble methods for wind and solar power forecasting—a state-of-the-art review. Renew Sustain Energy Rev 2015;50:82–91.
[45] Breiman L. Bagging predictors. Mach Learn 1996;24(2):123–40.
[46] Sahu A, Runger G, Apley D. Image denoising with a multi-phase kernel principal component approach and an ensemble version. In: 2011 IEEE applied imagery pattern recognition workshop (AIPR). 2011.
[47] Freund Y, Schapire R, Abe N. A short introduction to boosting. J Japn Soc Artif Intell 1999;14(771–780):1612.
[48] Liu H, Tian HQ, Li YF, Zhang L. Comparison of four Adaboost algorithm based artificial neural networks in wind speed predictions. Energy Convers Manage 2015;92:67–81.
[49] Cioppa TM. Efficient nearly orthogonal and space-filling experimental designs for high-dimensional complex models; 2002. DTIC Document.
[50] Law AM. A tutorial on design of experiments for simulation modeling. In: Proceedings of the winter simulation conference. IEEE Press; 2014. p. 2014.
[51] McKay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables from a computer code. Technometrics 1979;21, 7.
[52] Iman RL, Conover W. Small sample sensitivity analysis techniques for computer models with an application to risk assessment. Commun Stat Theory Methods 1980;9(17):1749–842.
[53] Stein M. Large sample properties of simulations using Latin hypercube sampling. Technometrics 1987;29(2):143–51.
[54] Iman RL. Latin hypercube sampling. In: Encyclopedia of quantitative risk analysis and assessment; 2008.
[55] Holland JH. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press; 1975.
[56] Coelho LdS. A quantum particle swarm optimizer with chaotic mutation operator. Chaos Solitons Fractals 2008;37(5):1409–18.
[57] Jovanović RŽ, Sretenović AA, Živković BD. Ensemble of various neural networks for prediction of heating energy consumption. Energy Build 2015;94:189–99.
[58] Chiu SL. Fuzzy model identification based on cluster estimation. J Intell Fuzzy Syst 1994;2(3):267–78.