DOI: 10.1515/orga-2015-0012

DE GRUYTER OPEN

# Web Application for Hierarchical Organizational Structure Optimization – Human Resource Management Case Study

Davorin Kofjač, Blaž Bavec, Andrej Škraba

University of Maribor, Faculty of Organizational Sciences, Kidričeva cesta 55a, 4000 Kranj, Slovenia
davorin.kofjac@fov.uni-mb.si, andrej.skraba@fov.uni-mb.si, blazbavec@gmail.com

**Background and Purpose:** In a complex strictly hierarchical organizational structure, undesired oscillations may occur, which have not yet been adequately addressed. Therefore, parameter values, which define fluctuations and transitions from one state to another, need to be optimized to prevent oscillations and to keep parameter values between lower and upper bounds. The objective was to develop a simulation model of hierarchical organizational structure as a web application to help in solving the aforementioned problem.
**Design/Methodology/Approach:** The hierarchical structure was modeled according to the principles of System Dynamics. The problem of the undesired oscillatory behavior was addressed with deterministic finite automata, while the flow parameter values were optimized with genetic algorithms. These principles were implemented as a web application with JavaScript/ECMAScript.
**Results:** Genetic algorithms were tested against well-known instances of problems for which the optimal analytical values were found. Deterministic finite automata was verified and validated via a three-state hierarchical organizational model, successfully preventing the oscillatory behavior of the structure.
**Conclusion:** The results indicate that the hierarchical organizational model, genetic algorithms and deterministic finite automata have been successfully implemented with JavaScript as a web application that can be used on mobile devices. The objective of the paper was to optimize the flow parameter values in the hierarchical organizational model with genetic algorithms and finite automata. The web application was successfully used on a three-state hierarchical organizational structure, where the optimal flow parameter values were determined and undesired oscillatory behavior was prevented. Therefore, we have provided a decision support system for determination of quality restructuring strategies.

***Keywords:*** *hierarchical organizational structure, genetic algorithms, deterministic finite automata, system dynamics, optimization, human resources*

## 1 Introduction

Human resource management in larger organizations presents a complex problem that cannot be addressed properly without using the quantitative methods. One example of a complex human resources management problem is a hierarchical human resource problem, which can be found in the army. It is supposed that a person can be promoted from a lower to a higher rank, without skipping ranks and never in an opposite way. Therefore, such a problem represents a kind of "supply chain" in which a change in a particular element of the chain affects all subsequent elements.

The key issue that need to be addressed in the aforementioned problem are: a) the time variability of parameter limits, b) oscillations in the acquired strategies, and c) an understanding of the hierarchical model of human resources. The described problem has been addressed in the context of several studies (Mehlman, 1987; Grinold and Marshall, 1977, Vajda, 1978; Bartholomew et al., 1991; De Feyter, 2007; Huang et al., 2009), but the variable lim-

Received: March 8, 2015; revised: May 2, 2015; accepted May, 26, 2015

its and the undesired oscillations (Škraba et al., 2011) have not yet been adequately addressed.

The main objective of the research presented in this paper is to develop a web application that will enable the determination of strategies on a hierarchical model of human resources. When developing the application, we have used using genetic algorithms, finite automata, and modeling according to the principle of system dynamics. The application can be used to define strategies, which consider variable parameters' limits and undesired oscillations in the acquired strategies. The system was implemented as a web application with the JavaScript/ECMAScript programming language, and can be accessible worldwide. The system also addresses visualization, via which the optimization process should be displayed, as well as the results. The solution also aims to adequately explain complex concepts for educational purposes, the transparency of results, and is easy to use.

## 2 Hierarchical organizational structure model

In order to develop a web application for optimization of the hierarchical organizational structure, a model of the structure needed to be developed first. The model was developed using the principles of system dynamics (Forrester, 1973). System dynamics (SD) is used to understand the behavior of complex systems, usually over time.

In a hierarchical organizational structure, it is assumed that the transitions between different ranks (classes) are possible only from a lower to a higher rank without skipping ranks. We also assume that the degradation is not possible, however, an individual may leave the system (fluctuation). Ranks, i.e. classes, are represented with stocks, while the transitions and fluctuations are modeled with rates. The causal loop diagram of the model is presented in Fig. 1, while the corresponding system dynamics model, modeled with Powersim, is presented in Fig. 2.

Each state ($X1$, $X2$, $X3$) represents the number of people in a particular rank. Each state has an inflow and an outflow. Inflow $A$ to the state $X1$ represents the initial recruitment, while the inflows $R1$ and $R2$ to states $X2$ and $X3$ represent the transitions from the previous states (ranks) that at the same time represent the outflow from those states. Flow $A$ is represented in a tabular way, i.e. at each time step a different recruitment value is required. Flows $R1$, $R2$, and $R3$ are determined by the values of $X1$, $X2$, and $X3$ and coefficients from *R1_table*, *R2_table*, and *R3_table*, respectively. The coefficients in *R1_table*, *R2_table,* and *R3_table* are represented in a tabular way, i.e. at each time step a different transition coefficient value is required. Each state also has the fluctuation outflow ($F1$, $F2$, $F3$), where people depart the rank for different reasons (new job, retirement, etc.).

Flows *F1*, *F2*, and *F3* are determined by the values of $X1$, $X2$, and $X3$ and coefficients *F1_table*, *F2_table*, and *F3_table*, respectively. Similarly to the *Rx_table* coefficients, the *Fx_table* coefficients are represented in a tabular way, i.e. at each time step a different fluctuation coefficient value is determined. The coefficients in $A$, *Rx_table* and *Fx_table* are originally obtained from historic statistical data in order to model and analyze the current situation. However, during the optimization process, as explained later in the text, these coefficients are determined with genetic algorithms to ensure an optimal system response.

The *CSE* (Cumulative Square Error), and the connected elements, measure the integral of root mean squared error of deviation (distance) of the system's state ($X1$, $X2$, $X3$) from the desired state defined by *Z1_table*, *Z2_table*, and *Z3_table*, respectively, in which the desired values of $X1$, $X2$, and $X3$ at each time step are stored in a table. An example of tabular values for the elements *R1_table*, *F1_table*, and *Z1_table* is presented in Table 1.

There are six balancing loops in the model, which can be observed in Fig. 1. The loops interconnect the following elements: $F1$ and $X1$, $F2$ and $X2$, $F3$ and $X3$, $R1$ and $X1$, $R2$ and $X2$, and $R3$ and $X3$. The balancing loops are crucial for the system's behavior, and their role is to guide the system's state towards its desired state.

The following is the mathematical model of the hierarchical model in discrete time. For example, state vari-

*Table 1: An example of tabular values for the elements R1_table, F1_table and Z1_table*

| R1_table | |
|---|---|
| Time step | Value |
| 1 | 0.16 |
| 2 | 0.14 |
| 3 | 0.08 |
| 4 | 0.07 |
| 5 | 0.07 |
| 6 | 0.07 |

| F1_table | |
|---|---|
| Time step | Value |
| 1 | 0.1 |
| 2 | 0.01 |
| 3 | 0.01 |
| 4 | 0.02 |
| 5 | 0.03 |
| 6 | 0.03 |

| Z1_table | |
|---|---|
| Time step | Value |
| 1 | 85 |
| 2 | 80 |
| 3 | 75 |
| 4 | 75 |
| 5 | 75 |
| 6 | 75 |

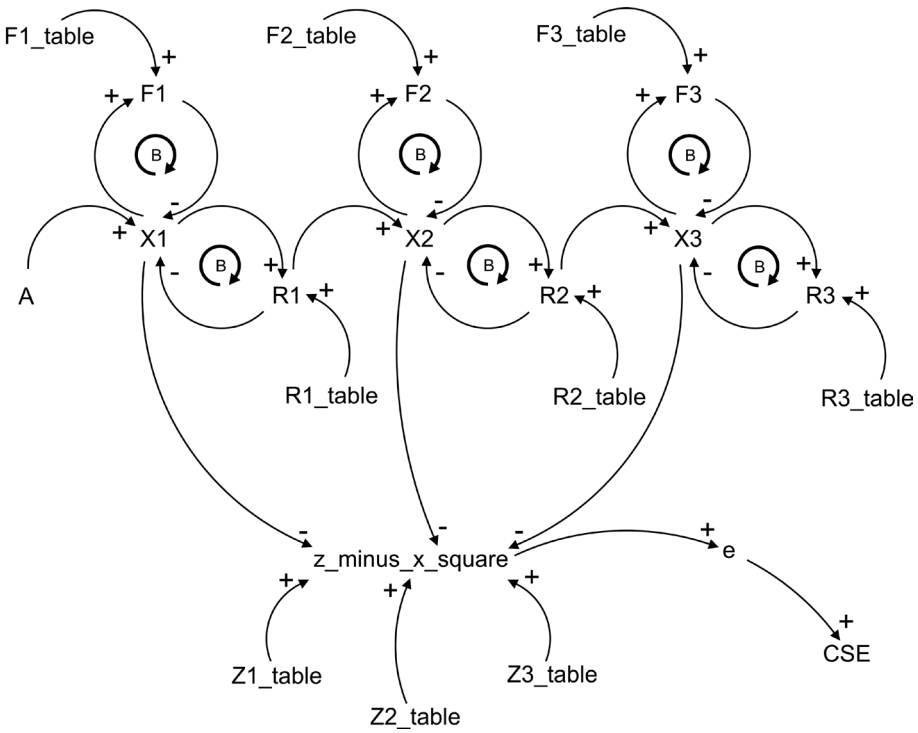


*Figure 1: Causal loop diagram of a hierarchical model with three states*

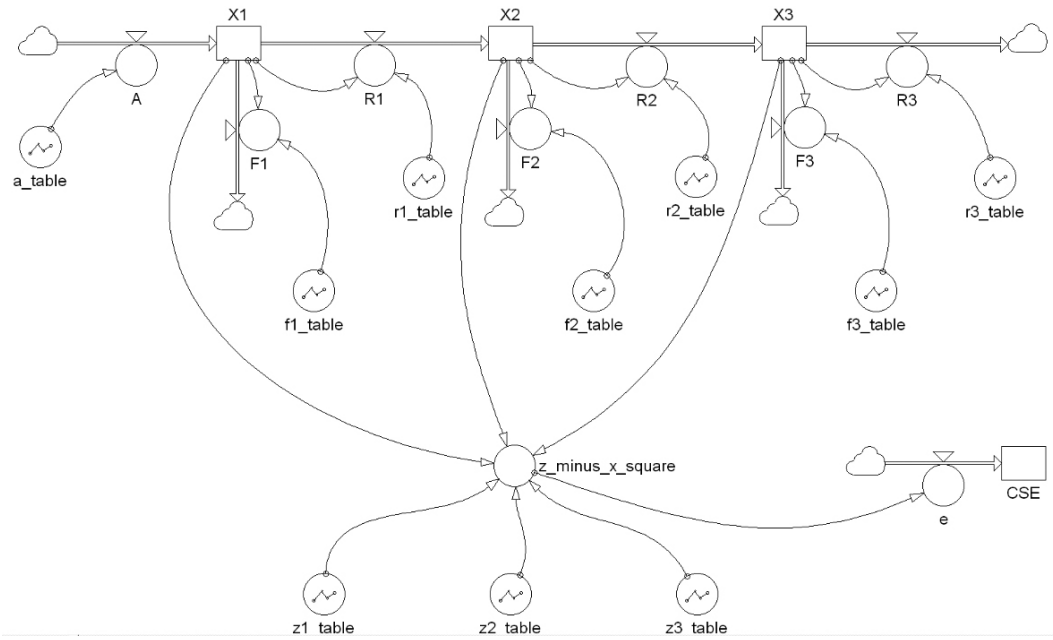


*Figure 2: Hierarchical model with three states modeled by the principles of system dynamics*

able $X1(k)$ represents the number of persons in the first rank at the current time $k$. In order to compute the value of the state variable in the next time step ($k + 1$), one must take into account the present state value $X1(k)$ to which the present inflows are added and from which the present outflows subtracted. In our case, the $A(k)$, as the present recruitment inflow variable, is added, while the present transition and fluctuation outflow variables, $R1(k)$ and $F1(k)$, are subtracted. The values of the state variables $X2$ and $X3$ are obtained similarly.

With the defined model, we can now formulate the problem that we address in this paper: to determine how to define recruitment, transitions between ranks and fluctuations to reach a new, i.e. the desire, organizational structure.

$$X1(k+1) = X1(k) + (A(k) - R1(k) - F1(k))\Delta t$$
$$X2(k+1) = X2(k) + (R1(k) - R2(k) - F2(k))\Delta t$$
$$X3(k+1) = X3(k) + (R2(k) - R3(k) - F3(k))\Delta t$$
$$CSE(k+1) = CSE(k) + e(k)\Delta t$$
$$X1(0) = X1\_init\_value$$
$$X2(0) = X2\_init\_value$$
$$X3(0) = X3\_init\_value$$
$$CSE(0) = 0$$
$$F1(k) = F1\_table(k) \cdot X1(k)$$
$$F2(k) = F2\_table(k) \cdot X2(k)$$
$$F3(k) = F3\_table(k) \cdot X3(k)$$
$$R1(k) = R1\_table(k) \cdot X1(k)$$
$$R2(k) = R2\_table(k) \cdot X2(k)$$
$$R3(k) = R3\_table(k) \cdot X3(k)$$
$$z\_minus\_x\_square(k) = (Z1\_table(k) - X1(k))^2 + (Z2\_table(k) - X2(k))^2 + (Z3\_table(k) - X3(k))^2$$
$$e(k) = z\_minus\_x\_square(k)$$
$$k = 0,1,2,...$$

# 3 Optimization methods

The aforementioned problem formulation, in which merely the minimization of the distance to the target function by considering the boundaries, is insufficient and may result in a possible undesired oscillatory solutions (Škraba et al., 2011). To overcome undesired oscillations, the Finite Automata (FA) was utilized. Further, genetic algorithms (GA) were used to define the optimal flow coefficients.

## 3.1 Finite automata

FA is an abstract machine that considers all the possible system states while taking into account a sequence of input symbols (Hopcroft et al. 2001). The transition between states occurs when a certain condition is fulfilled. The system controls the sequence of state transitions and identifies an illegal state, which may be used to trigger an event. The allowed states are called terminal states.

For our purposes, the deterministic finite automaton (DFA) is considered, which is used to solve complex problems, such as design and development of distributed simulation for evaluation of supply chains (Venkateswaran and Son, 2004), time-optimal coordination of flexible manufacturing systems (Kobetski and Fabian, 2009) and symbolic string analysis for vulnerability detection (Yu et al., 2014). The DFA in our example contains the following components:

- set of possible states $S = \{S_0, S_1, S_2, S_3, S_4, S_5\}$,
- comparison alphabet $CA = \{f, e, g\}$,
- initial state $i = S_0$,
- set of terminal states $T = \{S_0, S_1, S_2, S_3, S_4\}$,
- transition table δ.

An illegal state in our example is the $S_5$. If the DFA reaches this state during the simulation run, it means that the response of the system is following undesired oscillatory behavior. The undesired oscillatory behavior occurs if the system's response trajectory is going "up-down-up" or "down-up-down". If the DFA stops in any of the terminal states at the end of the simulation run, the response of the

*Table 2: The transition table δ for the DFA*

| State | Comparison alphabet | | |
|---|---|---|---|
| | f | e | g |
| $S_0$ | $S_2$ | $S_0$ | $S_1$ |
| $S_1$ | $S_3$ | $S_1$ | $S_1$ |
| $S_2$ | $S_2$ | $S_2$ | $S_4$ |
| $S_3$ | $S_3$ | $S_3$ | $S_5$ |
| $S_4$ | $S_5$ | $S_4$ | $S_4$ |
| $S_5$ | $S_5$ | $S_5$ | $S_5$ |

system is following the desired trajectory. An example of the transition table δ used in this research is presented in Table 2, while the DFA graph is shown in Figure 3. A more detailed explanation of the DFA can be found in Škraba et al. (2011).

### 3.2 Genetic algorithms

To determine the flow parameters (recruitment, transitions, fluctuations) in the model, genetic algorithms (GA) were used. GA belong to the evolutionary algorithms, which are used for solving complex optimization problems, e.g. optimization of manpower in hierarchical systems (Škraba et al., 2015), optimization of the health care system (Steiner et al., 2015), design of a production strategy (Mitsuyuki et al., 2014), or optimization in production scheduling (Kofjač and Kljajić, 2008).

GA are used when optimal solutions are not known, and the user is satisfied with near optimal solutions. GA are based on natural selection where the fittest individuals can survive (Gen and Cheng, 1997). The individuals in GA are represented by a chromosome. A chromosome is composed of genes. The next generation of individuals is selected by a selection strategy from the previous generation, and the selected individuals are subject to crossover and mutation operations. Crossover and mutation operations are used to produce new individuals from the existing ones. Crossover takes two parents and produces a new individual, while the mutation operation produces new individuals by changing genes of one individual. The fitness of an individual is assessed by a criteria function.

Let us present the implementation of GA in our case. The chromosome is encoded with a binary representation. The chromosome is composed of several sub-chromosomes; each sub-chromosome represents a value of a particular flow element. The number of genes in a particular sub-chromosome is dependent on the lower and upper boundaries of the flow element. The larger the interval, the more genes are needed to represent those values with binary encoding. The candidate chromosomes are selected with a roulette wheel selection. Furthermore, elitism is utilized to carry the best *n* chromosomes into the next generation. The mutation utilized in our implementation is bit-flip at random places. The crossover operator used here is a one-point crossover. The fitness function in our case is represented as an integral of the root mean squared error of deviation of the system's state from the desired state.

## 4 Web application

The web application was implemented using the HTML and JavaScript programming languages, which enables the development of solutions for different platforms. The web application was developed on a Windows 7 platform running a WAMP server. JavaScript is an interpreted programming language with object-oriented capabilities (Flanagan, 2006). Almost every desktop computer, tablet or smartphone has a JavaScript interpreter, thus making this programming language ubiquitous. Also critical is that every Internet browser supports JavaScript, thus enabling the solutions developed in JavaScript to be accessible virtually from anywhere.

The architecture of the proposed web application is presented in Fig. 4. The user interface was implemented with HTML5/CSS. The main web application (data input and results output) is implemented with JavaScript to ensure real-time output of optimization results via tables and graphs. The main application then calls the Optimization module, providing it with input parameters for optimization. On the basis of input parameters, the Optimization module interacts with the GA module, the DFA module, and the SD Model module. The results of optimization are then reported back to the main web application by the Optimization module and displayed on the user interface.

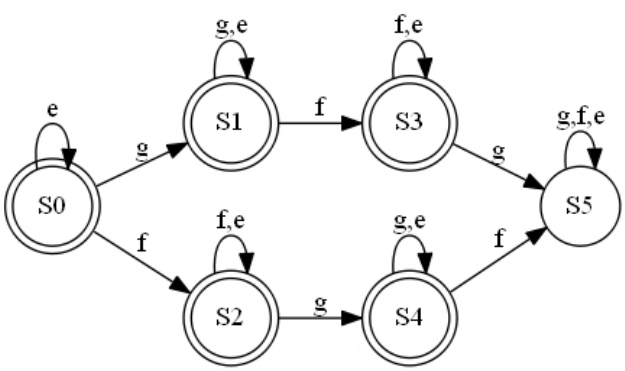


*Figure 3: The representation of the DFA as a graph*

## 5 Results

The GA were tested and validated with several well-known problem instances, for which the analytical solutions are known, such as:

- Elliptic paraboloid:

$$z(x,y) = x^2 + y^2 + 10, \text{ where } \max z(0,0) = 10$$

- 3D Wave:

$$z(x,y) = -\left(0.5 + \frac{\sin\left(\sqrt{2x^2 + 2y^2}\right)^2 - 0.5}{1 + 0.1\left(x^2 + y^2\right)}\right), \text{ where } \max z(0,0) = 0$$

- Rastrigin's function:

$$z(x,y,p,q) = 10 - 20\left(\left(x+p\right)^2 - 10\cos\left(2\pi\left(x+p\right)\right)\right) + \left(\left(y-q\right)^2 - 10\cos\left(2\pi\left(y+q\right)\right)\right),$$

$$\text{where } \max z\left(-\frac{1}{3}, -\frac{1}{4}, \frac{1}{3}, \frac{1}{4}\right) = 410$$

To test GA on the aforementioned instances, GA were initialized with the following values:

- Population size: 100,
- Mutation rate: 0.5,
- Crossover rate: 1,
- Maximum number of generations: 500,
- Stopping condition: maximum number of generations.

The GA were able to reach an optimal solution for the elliptic paraboloid and the 3D wave function, and a near-optimal solution for the Rastrigin's function within 0.0001% deviation, due to the resolution of the binary encoding. Therefore, we can conclude that GA were verified and validated successfully.

The web application was tested on the hierarchical model described earlier. The recruitment, inflow and outflow boundaries were set to [5, 8], [0.01, 0.16], [0.01, 0.1], respectively. The initial values for states *X*1, *X*2, and *X*3 were set to 100, 70, and 50, respectively. The desired values for the aforementioned states were set to 75, 95, and 40, respectively. The goal was to determine such recruitment, inflow and outflow rates that the system would reach the desired state values. In order to test the GA on the hierarchical model, the following settings were used:

- Population size: 100,
- Mutation rate: 0.01,
- Crossover rate: 1,
- Maximum number of generations: 500,
- Stopping condition: maximum number of generations.

The optimization results without DFA and with DFA are presented in Fig. 5 and Fig. 6, respectively. Coulours are visible in the internet version of the paper available from http://dx.doi.org/10.1515/orga-2015-0012.

Each graph is presented by two axes. X-axis (blue line) presents the number of simulation steps, while the y-axis (green and/or red line) shows the value of state and/or flow elements. For example, if observing the upper left graph in Fig. 5, number 0 presents the origin, while the numbers 5 and 100 show the maximum value on the x-axis and y-axis, respectively. The curve marked with a red color shows the desired trajectory defined by *Z*1, while the green curve presents the simulated trajectory of the state element *X*1. One can observe, that the trajectory of *X*1 is closely following the desired trajectory *Z*1, meaning that the web application has successfully optimized the *X*1 trajectory.

The upper part of the figures presents the trajectories for state elements *X*1, *X*2, and *X*3 and their corresponding desired trajectories represented by *Z*1, *Z*2, and *Z*3. The middle part shows the trajectories of transition flow elements *R*1, *R*2, and *R*3. The bottom part of both figures

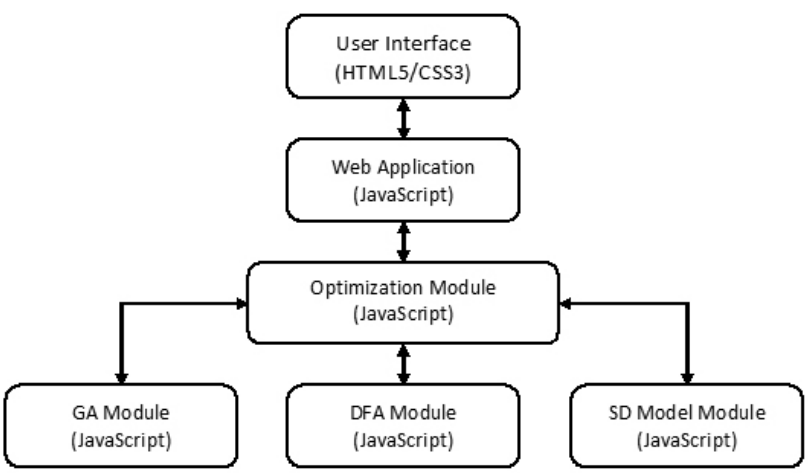


*Figure 4: The architecture of the web application*

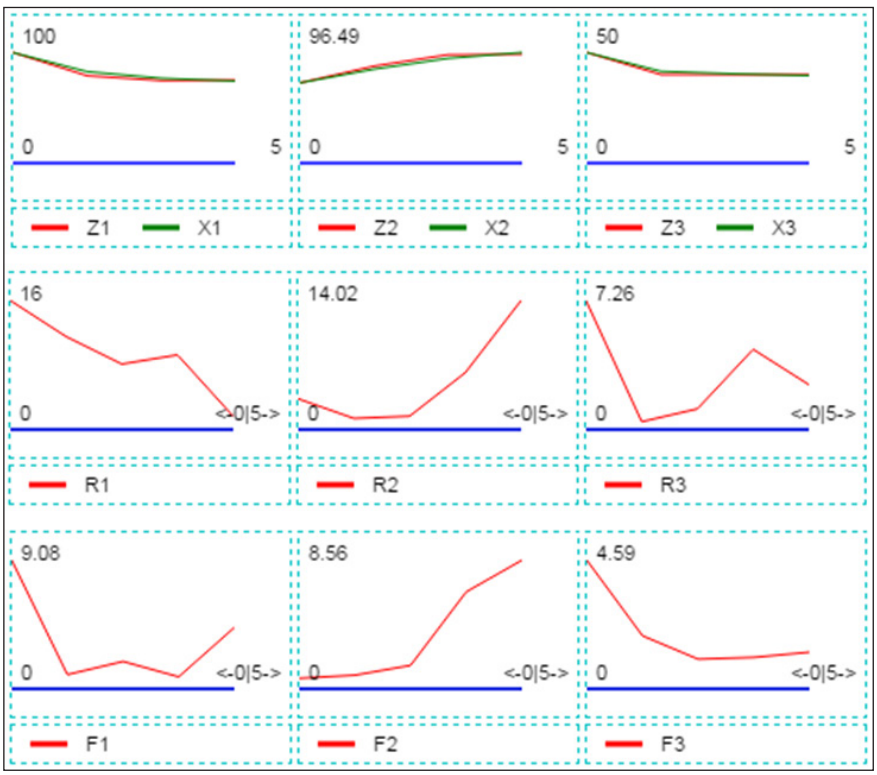


*Figure 5: A comparison of trajectories of state and flow elements achieved without DFA*

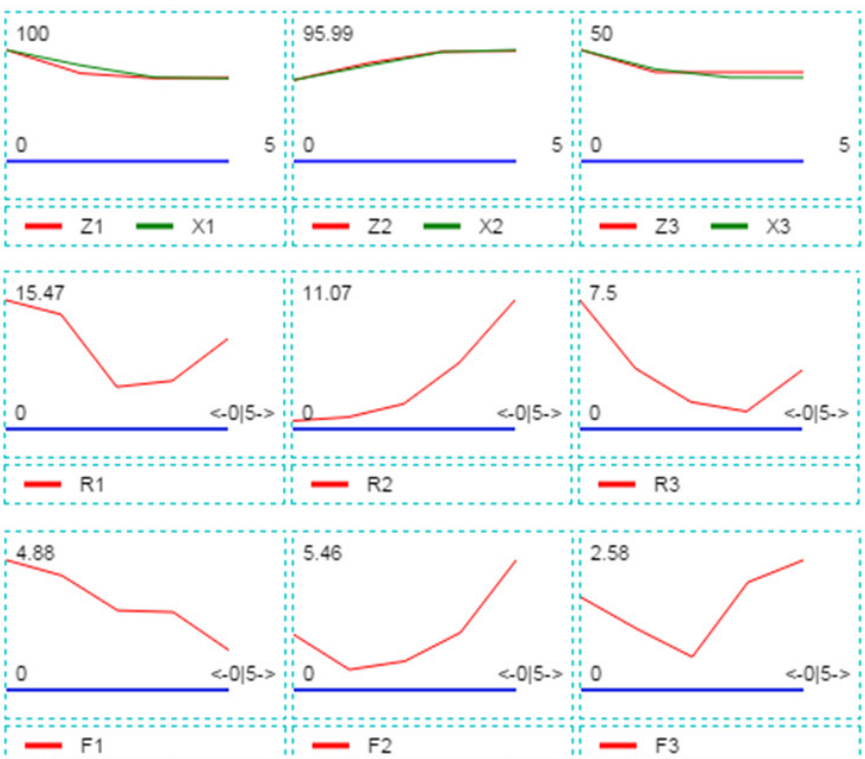


*Figure 6: A comparison of trajectories of state and flow elements achieved with DFA*


*Figure 7: Web-based application displayed on a mobile device. Coulours are visible in the internet version of the paper available from http://dx.doi.org/10.1515/orga-2015-0012*

presents the resulting trajectories for the fluctuation flow elements $F1$, $F2$, and $F3$. Both optimizations, with and without DFA, achieve the desired values for the state elements $X1$, $X2$, and $X3$. However, the desired state values are achieved in a different way. When observing the results for the flow elements in Fig. 5, one can observe there are undesired oscillations for the $R1$, $R3$, and $F1$ elements, due to the fact of the DFA not being used. In contrast, the use of DFA results in trajectories of flow elements without undesired oscillations, as shown in Fig. 6.

The example of the application can be observed in Fig. 7, where a system is run on an Android mobile device. The GUI consists of three sections: Simulation control, Output, and Graph/Table display.

Simulation control section of the GUI is used to input the following simulation parameters:

- Max. generations – the maximum number of generations produced by the GA.
- Mutation rate – the share of individuals in a population, which are mutated.
- Mutation type (0 or 1) – the type of mutation used in the mutation process.
- Population size – the number of individuals in a population for GA.
- FA – determines if finite automata is used in optimization.
- Table output – determines if simulation results of parameters for each generation are displayed.
- Sim. table output – determines if simulation results of parameters for each generation and each simulation step are displayed.
- Verbosity – determines if the results are output as text.
- Fitness func. output – determines if a fitness function values per generation are displayed.
- Best strategy – determines if only the simulation results of parameters for the best solution are displayed in the table.

Further, Simulation control section offers four buttons for simulation control:

- Start – starts the simulation run.
- Stop/Resume – pauses and resumes the simulation run.
- Step – performs one simulation step.
- Reset – clears the simulation results and initializes simulation parameters.

In the Output section, the current numerical results are displayed dynamically, including the current GA generation and simulation step. The start time and the elapsed simulation time are also shown. Finally, the main simulation results are shown: current fitness function value and current level variable values for $X1$, $X2$, and $X3$.

The bottom part of GUI presents the simulation results via graphs and tables. The values presented in graphs are time dependent. The first graph presents the change of the best fitness function value over time. The following graph presents the CSE value over time. Following are the graphs where the comparison of state variables (green line) with their desired values (red line) is shown. Next, graphs with rate element values and rate coefficients are presented, respectively. Finally (not shown in Fig. 7), the numerical simulation results are presented in tables.

## 6 Conclusion

We have successfully implemented the web application for hierarchical organizational structure optimization and tested it on a three-state hierarchical human resources structure. All components of the web application, including the hierarchical organizational structure model, genetic algorithms and finite automata, were implemented with the JavaScript programming language.

First, the JavaScript System Dynamics modeling library was developed, thus enabling simple and efficient implementation of System Dynamics models with any desired number of states and flows. Next, the hierarchical organizational structure model was developed by the principles of System Dynamics. The genetic algorithms were successfully tested on well-known problem instances, where the analytical solution is known, and used to optimize the hierarchical model flow parameters. The finite automata method was used to prevent oscillations that might occur at transitions between structure's states.

With the developed application, the user is able to optimize the flow parameters, i.e. define strategies, where dynamic parameter boundaries are considered as well as oscillations in the acquired strategies. By applying such approach, it is possible to keep the organizational structure stable and robust. Based on the abovementioned, the developed web application represents an approach, which can be used in the restructuring of large hierarchical organizational systems, such as the armed forces.

Because the system is implemented with JavaScript as a web platform, it can be used on mobile and other devices, enabling to control the organizational structure anywhere around the globe. The user is not limited to use the application only on a desktop computer in an office and is able to analyze the organizational structure's current state and quickly respond to possible challenges at any time.

In the future, our goal is to test the application on an organizational structure with more states in order to determine the impact of an increased number of states, and consequently flow parameters, on computational time. Application implemented with JavaScript programming language might prove computationally costly, because JavaScript has been traditionally implemented as an interpreted language. Applications implemented with interpreted languages are typically slower than the ones that need to be compiled. Finally, because the model presented in this paper is highly similar to a supply chain structure, we want

to extend its application to similar structures.

## Acknowledgment

This research was supported by the Ministry of Higher Education, Science and Technology of the Republic of Slovenia (Contract No. P5-0018).

## References

Bartholomew, D.J., Forbes A.F., & McClean, S.I. (1991). *Statistical techniques for manpower planning.* Chichester: John Wiley & Sons.

De Feyter, T. (2007). Modeling mixed push and pull promotion flows in manpower planning. *Annals of Operations Research*, 155(1), pp. 22 – 39, http://dx.doi.org/10.1007/s10479-007-0205-1

Flanagan, D. (2006). *Javascript: The definitive guide – 5th edition*. Cambridge, MA: O'Reilly Media.

Forrester, J. (1973). *Industrial Dynamics*. Cambridge, MA: MIT Press.

Gen, M., & Cheng, R. (1997). *Genetic Algorithms & Engineering Design*. Chichester: John Wiley & Sons, Inc.

Grinold, R.C., & Marshall, K.T. (1977). *Manpower planning models.* New York, NY: Elsevier North-Holland.

Hopcroft, J.E., Motwani, R., & Ullman, J.D. (2001). *Introduction to Automata Theory, Languages, and Computation (2 ed.)*. Reading, MA: Addison Wesley.

Huang, H., Lee, L., Song, H., & Eck, B.T. (2009). SimMan - A simulation model for workforce capacity planning. *Computers & Operations Research*, 36(8), pp. 2490 – 2497, http://dx.doi.org/10.1016/j.cor.2008.10.003

Kobetski, A, & Fabian, M, (2009). Time-Optimal Coordination of Flexible Manufacturing Systems Using Deterministic Finite Automata and Mixed Integer Linear Programming. *Discrete Event Dynamic Systems - Theory and Applications*, 19(3), pp. 287-315, http://dx.doi.org/10.1007/s10626-009-0064-9

Kofjač, D., & Kljajić, M. (2008). Application of genetic algorithms and visual simulation in a real-case production optimization. *WSEAS transactions on systems and control*. 3(12), pp. 992-1001.

Mehlman, A. (1980). An approach to optimal recruitment and transition strategies for manpower systems using dynamic programming. *Journal of Operational Research Society*, 31(11), pp. 1009 – 1015, http://dx.doi.org/10.2307/2581281

Mitsuyuki, T., Hiekata, K., & Yamato, H. (2014). Design of production strategy considering the cutting peak demand of electricity in the shipbuilding industry. *Journal of Marine Science and Technology*, 19(4), pp. 425-437, http://dx.doi.org/10.1007/s00773-014-0261-6

Steiner, M.T.A., Datta, D., Neto, P.J.S., Scarpin, C.T., & Figueira, J.R. (2015). Multi-objective optimization in partitioning the healthcare system of Parana State in Brazil. *Omega - International Journal of Management Science*, 52, pp. 53-64, http://dx.doi.org/10.1016/j.omega.2014.10.005

Škraba, A., Kljajić, M., Papler, P., Kofjač, D., & Obed, M. (2011). Determination of recruitment and transition strategies. *Kybernetes*, 40(9/10), pp. 1503-1522, http://dx.doi.org/10.1108/03684921111169512

Škraba, A., Kofjač, D., Žnidaršič, A., Maletič, M., Rozman, Č., Semenkin, E.S., Semenkina, M.E., & Stanovov, V.V. (2015). Application of Self-Configuring genetic algorithm for human resource management. *Journal of Siberian Federal University - Mathematics and Physics*. 8(1), pp. 94-103.

Vajda, S. (1978). *Mathematics of Manpower Planning*. Chichester: John Wiley & Sons.

Venkateswaran, J., & Son, Y.J. (2004). Design and development of a prototype distributed simulation for evaluation of supply chains. *International Journal of Industrial Engineering - Theory Applications and Practice*, 11(2), pp. 151-160.

Yu, F., Alkhalaf, M., Bultan, T., & Ibarra, O.H. (2014). Automata-based symbolic string analysis for vulnerability detection. *Formal Methods in System Design*. 44(1), pp. 44-70, http://dx.doi.org/10.1007/s10703-013-0189-1

**Davorin Kofjač** obtained his Ph.D. from the University of Maribor in the field of Information systems management. He is a researcher and an assistant professor at the University of Maribor, Faculty of Organizational Sciences at the Cybernetics and DSS Laboratory. His main research interests include modelling and simulation, decision support systems, operational research and artificial intelligence. He was involved in many EU, NATO, bilateral and national projects and is an author of more than 120 publications in international journals, monographs and conferences. He is a member of ACM, INFORMS and SLOSIM.

**Blaž Bavec** gained his M.Sc. from the University of Maribor in the field of complex support system management creation. He finished his degree, while he was working on several projects at the Cybernetics and DSS Laboratory at the University of Maribor, Faculty of Organizational Sciences. His main research interests include systems' creation, implementation and management.

**Andrej Škraba** obtained his Ph.D. in the field of Organizational sciences from the University of Maribor. He works as a researcher and an associate professor at the University of Maribor, Faculty of Organizational Sciences in the Cybernetics and DSS Laboratory. His research interests cover modeling and simulation, systems theory and decision processes. He is a member of the System Dynamics Society and SLOSIM.