Knowledge-based Solution Construction for Evolutionary Minimization of Systemic Risk

Krzysztof Michalak

Department of Information Technologies,
Institute of Business Informatics,
Wroclaw University of Economics, Wroclaw, Poland
krzysztof.michalak@ue.wroc.pl

Abstract. This paper concerns a problem of minimizing systemic risk in a system composed of interconnected entities such as companies on the market. Systemic risk arises, when, because of an initial failure of a limited number of elements, a significant part of the system fails. The system is modelled as a graph, with some nodes in the graph initially failing. The spreading of failures can be stopped by protecting nodes in the graph, which in case of companies can be achieved by setting aside reserve funds. The goal of the optimization problem is to reduce the number of nodes that eventually fail due to connections in the system.
This paper studies the possibility of utilizing external knowledge for solution construction in this problem. Rules representing reusable information are extracted from solutions of problem instances and are used when solving new instances.
Experiments presented in the paper show that using rule-based knowledge representation for constructing initial population allows the evolutionary algorithm to attain better results during the optimization run.

Keywords: Knowledge-based optimization, Rule-based knowledge representation, Graph problems, REDS graphs

1 Introduction

Many real-life phenomena such as bankruptcies, epidemics, viruses in computer networks, failures in power grids and wildfires can be modelled in terms of failures of interconnected nodes in a network. Once a node has failed (e.g. a person fell ill or a company has gone bankrupt, etc.) it incurs an additional load on surrounding nodes (e.g. exposing acquaintances to germs, delaying payments to related companies, etc.). Because of that increased load, surrounding nodes may also fail giving rise to a cascading failure which can be a catastrophic event destroying the entire system.

The problem tackled in this paper is a problem of reduction of systemic risk [1]. Systemic risk is often analyzed using network models [2, 3]. As in other works on spreading of threats in networks, in this paper the system is modelled as an undirected graph $G = \langle V, E \rangle$ with N_v nodes. The nodes in the graph represent

the entities subject to failures and they can be in one of the states 'F' - failed, 'P' - protected, or 'U' - unprotected. In the initial state S_0 some nodes from a given non-empty set $\emptyset \neq S \subset V$ are set to the state 'F' and the remaining ones are set to the state 'U'. In the problem studied here each node in the graph can be protected or not, but the total resources that can be used for protecting the graph are limited and we assume that at most N_t nodes can be protected. Therefore, a problem instance is an ordered triple $\langle G, S_0, N_t \rangle$, where N_t is the total number of nodes that can be protected in the graph. The search space is $\{0,1\}^{N_v}$ with the constraint that the number of ones in any solution cannot exceed N_t. The value of the objective function for a solution $x \in \{0,1\}^{N_v}$ is calculated by simulating the spreading of failures in the graph. Initially, nodes corresponding to the elements $x[i] = 1$ are protected by setting their state to 'P'. In each time step $t > 0$ failures progress from nodes in the 'F' state to adjacent nodes in the 'U' state. Nodes in the 'P' state are immune to failures, and this protection lasts to the end of the simulation. When failures stop spreading, the solution x is evaluated by calculating how many nodes in the graph survived (i.e. are in the state 'U' or 'P').

In this paper an optimization algorithm finds solutions that are vectors, which indicate which nodes in the graph to protect. The proposed approach takes advantage of the fact that a solution for the entire graph can be decomposed to decisions for individual nodes (*protect/do not protect*). The presented method improves solutions in an initial population of an evolutionary algorithm by making decisions for individual nodes using a machine learning algorithm. Initial populations constructed in this way lead to better optimization results than randomly initialized populations.

2 Proposed Approach

The approach proposed in this paper is to use rules for building better initial solutions for an evolutionary optimizer. The rules used in this paper were obtained from solutions of the studied problem found for instances with a low number of nodes in the graph. The proposed approach is summarized below.

1. Run an optimization algorithm on a training set containing several (e.g. $N_{runs} = 30$) instances of the problem
2. Build a training data set in the following manner:
   - For each node $v \in V$ calculate its attributes based on the structure of the graph and on the initial state S_0. For the list of attributes used in this paper see Table 1.
   - From each of the 30 runs of the optimizer get the best solution found and for each node create one object in the training data set with the attributes of the node and a class indicating if the node is protected in the analyzed solution. This analysis results in N_v objects generated for the training sample, where N_v is the number of nodes in the graph.

- Agglomerate the objects generated in all N_{runs} runs of the test, resulting in a training sample containing $N_{runs} \cdot N_v$ objects partitioned into two classes (*protect* and *do not protect*).

3. Optionally, perform a feature selection step. The goal of this step is twofold. First, with a correctly selected set of attributes classification algorithms may perform better than on the full attribute set. Second, feature selection methods attempt to assess the importance of the attributes and, in some cases, provide a ranking of the attributes which can be used to tell which attributes of the graph nodes are more and which less important.
4. Generate a set of rules using a chosen machine learning algorithm.
5. Use the obtained rules for generating initial solutions when solving other, possibly larger, problem instances.

3 Experiments and results

The experiments performed in this paper were conducted following the steps presented in Section 2.

3.1 Test instances

The graphs used in the experiments were REDS graphs [4] generated on a $[0,1] \times [0,1]$ square. The REDS graphs combine spatial relationships with social synergy effects. The edges consume "energy" of the nodes proportionally to the edge length D_{ij}, but the edge cost is discounted by the factor of $\frac{1}{1+Sk_{ij}}$ for nodes v_i and v_j that have k_{ij} neighbours in common. This method of generating graphs results in a distribution of edges with denser cliques separated by relatively sparse areas, which is a distribution that resembles that of a real-life social network [4]. The first round of experiments was focused on generating decision rules and in this round graphs with $N_v = 1000$ were generated with the maximum distance between connected nodes $R = 0.1$, the social energy $E = 0.15$ and the synergy parameter $S = 0.5$. In the second round of experiments in which the universality of the generated rules was tested, problem instances with $N_v = 1000, \ldots, 3000$ generated with parameters presented in Table 4 were used. Note, that all the parameters shown in Table 4 are adjustable, except for the average degree $\overline{k}$ which is calculated from the obtained graphs. The initial state S_0 was generated for each instance by setting $N_b = 50$ randomly selected nodes to the 'F' state. The choice of the initially failed nodes was kept constant between different runs of the tested methods on the same graph in order to allow comparing the results. The above-described settings were confirmed in preliminary experiments to generate problem instances neither too easy (with failures immediately contained) nor too difficult (with almost the entire graph failing).

3.2 The optimization algorithm

The solution space of the systemic risk minimization problem presented in this paper is $\{0,1\}^{N_v}$ with a constraint limiting the number of the elements equal

to 1 to at most N_t (representing the limitation of the available resources used for protecting graph nodes). To solve this problem without additional repair operators or constraint handling techniques an evolutionary algorithm was used with the chromosomes being elements of $[0,1]^{N_v}$. In order to calculate the objective function value, each solution $s \in [0,1]^{N_v}$ was decoded by selecting N_t largest elements and setting them to 1 and the remaining elements to 0, obtaining a vector $s' \in \{0,1\}^{N_v}$. Subsequently, a simulation of the spreading of failures in the graph was performed starting with the initial state S_0. Nodes corresponding to positions equal to 1 in s' were set to the 'P' state, excluding the nodes already failed in the initial state S_0. This exclusion was imposed in order to prevent the algorithm from finding an easy solution of protecting all the nodes set to 'F' in S_0 thereby preventing the failures from occurring at all.

The population size was set to $N_{pop} = 1000$ and the algorithm was allowed to run for a maximum of $N_{gen} = 1000$ generations. In this paper multiple crossover and mutation operators were used, because the effectiveness of different operators may vary during the optimization run [5]. Multi-operator evolutionary algorithms have been found to work well for real-valued [6] and combinatorial optimization problems [7]. This approach also worked well for the Firefighter Problem (FFP) [8], which is a related optimization problem concerning the spreading of a threat ("fire") in a graph. Three classical crossover operators working on the entire chromosomes were used: single point, two-point and uniform crossover. Also, the Simulated Binary Crossover (SBX) [9] was used with the distribution index $\eta_c = 20$, working at each position of the chromosomes separately. The probability of performing the crossover operation on a pair of parent solutions was set to $P_{cross} = 1.0$. Five mutation operators working on the entire chromosomes were used: displacement, insertion, inversion, scramble and transpose with the probability of applying the mutation $P_{mut/s} = 0.05$ per specimen. Also, two mutation operators were used working at each position of the chromosomes separately: polynomial mutation [10] with the distribution index $\eta_m = 20$ and uniform mutation with the probability of applying the mutation $P_{mut/p} = \frac{1}{N_v}$ per position in the chromosome. For deciding which operator to select, an auto-adaptation mechanism was used, selecting the operators with probabilities proportional to operator success rates (i.e. the number of improved solutions generated by the operator, divided by the number of times the operator was used) [8]. In order not to exclude any operator completely from the operator selection process the minimum probability of selecting an operator using the auto-adaptation mechanism was set to $P_{min} = 0.05$. Parent selection was performed using a binary tournament.

3.3 Rule extraction and new solutions construction

For generating rules a training data set has to be prepared, based on the best solutions obtained in a number of runs of the optimization algorithm. In this data set the nodes of the graph are described by attributes and for each node a desired class is assigned (either *protect* or *do not protect* depending on whether the node has actually been protected in a given solution). Using the training data

set a machine learning algorithm builds rules which can subsequently be used for deciding if a given node in the graph should be protected or not. In the experiments the attributes presented in Table 1 were used.

Table 1: Attributes of a graph node $v \in V$ used in the experiments

Name	Description
failed	Indicates whether the node is set to the 'F' state at the beginning of the simulation (*failed* = 1) or not (*failed* = 0).
deg	Node degree. The number of edges that are adjacent to the node.
clos_centr	Closeness centrality. An inverse of the mean shortest path length from v to every other node in the graph.
betw_centr	Betweenness centrality. A measure based on the number of the shortest paths between other nodes that go through v.
fail_min_dist	The length of the shortest path between the given node and any node that is initially in the state 'F'.
fail_avg_dist	The average length of the shortest paths between the given node and all the nodes that are initially in the state 'F'.

Based on these attributes, rules were generated using the PART rule discovery algorithm [11] implemented in Weka 3.8.1 [12] using the default parameter settings. Rules discovered using all the attributes are presented in Figure 1. Numbers in parentheses denote how many training examples matched the rule and how many among those were misclassified. For classifying a node the rules are checked from top to bottom and the first matching rule is invoked generating a decision to either *protect* the node or *do not protect*. The last rule which does not have any conditions before the $\Longrightarrow$ sign is the default classification rule invoked when no other rules matched a given node.

The rules described above can be used to improve initial populations used by a population-based optimizer containing solutions represented as $[0, 1]^{N_v}$ vectors. Each solution in the initial population is modified with a probability $P_{mod/s}$. If a given solution is selected for modification, then each of the nodes in the solution is modified with a probability $P_{mod/p}$. A classification is performed using the rules, and if the classification result is *do not protect* the corresponding element in the solution is set to 0 and if the classification result is *protect* the element is set to 1. Note, that solution vectors contain values in the $[0, 1]$ range and they are decoded to values from the set $\{0, 1\}$ as described in Section 3.2. Therefore, even a node classified as *protect* by the classifier may not actually be protected if the N_t limit is exceeded.

To verify if the discovered rules allow generating good solutions, the same evolutionary algorithm was run again, but on a new set of 30 test instances generated separately from the training ones. In these test runs the algorithm started with the same initial populations without modifications and with initial populations improved using the procedure described above with both the probability

R01: *failed* ≤ 0 **and** *fail_min_dist* > 2 $\Longrightarrow$ *do not protect* (11686/833)

R02: *failed* > 0 $\Longrightarrow$ *do not protect* (1500/0)

R03: *betw_centr* ≤ 0.005532 **and** *clos_centr* ≤ 0.066908 **and** *deg* > 22 $\Longrightarrow$ *do not protect* (445/35)

R04: *fail_min_dist* > 1 $\Longrightarrow$ *do not protect* (8185/1989)

R05: *deg* > 18 $\Longrightarrow$ *do not protect* (3562/884)

R06: *betw_centr* > 0.001892 **and** *deg* ≤ 10 $\Longrightarrow$ *protect* (1603/628)

R07: *betw_centr* ≤ 0.00003 **and** *deg* > 6 $\Longrightarrow$ *do not protect* (90/9)

R08: *betw_centr* ≤ 0.003689 **and** *deg* > 6 **and** *deg* ≤ 16 $\Longrightarrow$ *do not protect* (109/17)

R09: *betw_centr* > 0.004271 **and** *deg* > 14 $\Longrightarrow$ *do not protect* (1106/479)

R10: *betw_centr* > 0.004271 **and** *deg* > 12 $\Longrightarrow$ *do not protect* (598/291)

R11: *betw_centr* > 0.004087 **and** *deg* ≤ 12 $\Longrightarrow$ *protect* (527/249)

R12: *# the default rule* $\Longrightarrow$ *do not protect* (589/210)

Fig. 1: Rules discovered using all the attributes.

$P_{mod/s}$ and the probability $P_{mod/p}$ varied in the range $\{0.2, 0.4, 0.6, 0.8\}$. From each of the 30 runs the objective function value of the best solution found was recorded. Table 2 presents medians from 30 runs obtained for each pair of values of $P_{mod/s}$ and $P_{mod/p}$ and obtained when no improvement of initial populations was performed (denoted "None" in the table). In order to verify significance of the results, statistical testing was performed using the Wilcoxon signed rank test [13]. In cases when a low p-value was obtained it could be concluded that the difference between median results produced by two compared methods was significant. In Table 2 the results significantly better than those obtained without rule-based improvement (at the significance level $\alpha = 0.05$) are marked in bold.

Clearly, when high probabilities of modifying a solution $P_{mod/s}$ and modifying individual positions $P_{mod/p}$ were set, good initial populations were obtained. In particular for $P_{mod/s} = 0.8$ and $P_{mod/p} = 0.8$ the largest median objective function value was obtained and the result was statistically significant.

3.4 Experiments with feature selection

A round of experiments described in this section was carried out using Weka in order to verify if the results could be further improved by selecting a good attribute set. Two methods of feature selection were tested. The first was to

Table 2: Medians from 30 runs obtained for each pair of values of $P_{mod/s}$ and $P_{mod/p}$. The best parameter setting is $P_{mod/s} = 0.8$ and $P_{mod/p} = 0.8$.

Modification probabilities	Median obj. func. value	p-value
$P_{mod/s} = 0.2$		
$P_{mod/p} = 0.2$	648.5	0.9836
$P_{mod/p} = 0.4$	653.0	0.7189
$P_{mod/p} = 0.6$	651.5	0.8797
$P_{mod/p} = 0.8$	648.5	0.0587
$P_{mod/s} = 0.4$		
$P_{mod/p} = 0.2$	643.5	0.7132
$P_{mod/p} = 0.4$	651.5	0.5440
$P_{mod/p} = 0.6$	637.0	0.5739
$P_{mod/p} = 0.8$	**659.0**	**0.0359**
None	647.5	-

Modification probabilities	Median obj. func. value	p-value
$P_{mod/s} = 0.6$		
$P_{mod/p} = 0.2$	663.5	0.3440
$P_{mod/p} = 0.4$	632.5	0.7971
$P_{mod/p} = 0.6$	649.5	0.4555
$P_{mod/p} = 0.8$	636.5	0.2755
$P_{mod/s} = 0.8$		
$P_{mod/p} = 0.2$	655.5	0.5439
$P_{mod/p} = 0.4$	655.0	0.0878
$P_{mod/p} = 0.6$	**667.5**	**0.0042**
$P_{mod/p} = 0.8$	**668.0**	**0.0015**

follow the filter approach and rank the attributes with respect to the Information Gain Ratio (IGR) measure [14]. Feature ranking obtained using this method is presented in Table 3. In order to select attributes one has to select those that are located on the top of the ranking. Building of rules based on three or fewer attributes from the ranking resulted in a rule set containing only one rule unconditionally returning *do not protect* for all nodes and the entire data set has six attributes. Therefore, two feature sets were considered: $FS_{rank,4} = \{$ *failed*, *fail_min_dist*, *betw_centr*, *deg* $\}$ and $FS_{rank,5} = FS_{rank,4} \cup \{$*clos_centr*$\}$.

Table 3: Ranking of features by the Information Gain Ratio (IGR) measure

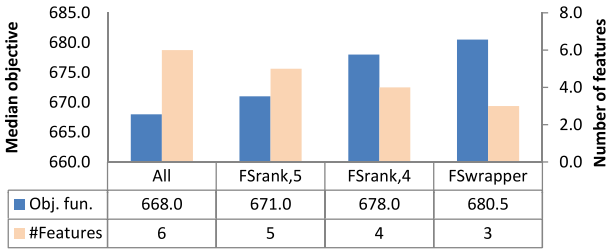
Rank	Information Gain Ratio	Feature name
1	0.05784	*failed*
2	0.03992	*fail_min_dist*
3	0.00927	*betw_centr*

Rank	Information Gain Ratio	Feature name
4	0.00913	*deg*
5	0.00271	*clos_centr*
6	0.00231	*fail_avg_dist*

Apart from the filter approach a wrapper feature selection was performed which selected the features: $FS_{wrapper} = \{$ *deg*, *betw_centr*, *fail_min_dist* $\}$. Attribute sets selected by the algorithms seem in line with what is known about the role of various node attributes in prevention of the spreading of threats. The attributes *deg* and *fail_min_dist* are used, for example, in heuristics described in the literature on the Firefighter Problem (FFP), which suggest defending nodes with high degrees and located close to fire first [15]. The *betw_centr* attribute value depends on how many shortest paths go through a node. Protecting nodes with high betweenness centrality should help preventing failures from spreading

quickly by following short paths in the graph. The *failed* feature is also important, because when *failed* = 1 the node is marked as failed in the initial state S_0 which makes it futile to try to protect that node.

Using the three feature sets $FS_{wrapper}$, $FS_{rank,4}$ and $FS_{rank,5}$ rule sets were built using the PART algorithm. These rule sets were used for improving initial populations with the modification probabilities set to the best found values $P_{mod/s} = 0.8$ and $P_{mod/m} = 0.8$. In Figure 2 the results obtained using feature selection are presented.



	All	FSrank,5	FSrank,4	FSwrapper
Obj. fun.	668.0	671.0	678.0	680.5
#Features	6	5	4	3

Fig. 2: Median best solutions obtained when initial population was improved using rules built on a set of attributes reduced using feature selection methods.

Attribute selection improved the results, with smaller attribute sets performing better than larger ones. The wrapper feature selection produced the best results, which is not very surprising, because wrappers tend to select attribute sets better suited for a particular learning algorithm than filters. On the other hand, wrappers usually require more computations, however, in presented experiments the wrapper feature selection took no more than several seconds to complete. Figure 3 presents rules discovered using the attributes selected by the wrapper feature selection method.

3.5 Testing the universality of the obtained rules

Experiments presented in this section were aimed at testing the universality of the rules discovered for $N_v = 1000$, by using them to generate initial solutions for instances with varying number of nodes. The tests were performed with $P_{mod/s} = 0.8$ and $P_{mod/p} = 0.8$, because these values turned out to be the best in the tests described in previous sections. Parameters of test instances used in this round of experiments are shown in Table 4.

Results presented in Table 5 are median values calculated over 30 runs on test instances of a given size obtained without improving the initial population (denoted "None" in the table) and obtained using rules presented in Figures 1 (denoted "All") and 3 (denoted "Wrapper"). For each test the medians obtained using rules are better than when the rules were not used. Results in the "All" and "Wrapper" columns significantly better than those in the "None" column

R01: *fail_min_dist* > 2 $\Longrightarrow$ *do not protect* (11686/833)

R02: *fail_min_dist* > 0 **and** *deg* ≤ 22 **and** *fail_min_dist* > 1
$\Longrightarrow$ *do not protect* (7191/1790)

R03: *fail_min_dist* > 0 **and** *deg* > 18 $\Longrightarrow$ *do not protect* (5001/1118)

R04: *fail_min_dist* > 0 **and** *deg* ≤ 10 **and** *betw_centr* > 0.001634
$\Longrightarrow$ *protect* (1625/634)

R05: *fail_min_dist* ≤ 0 $\Longrightarrow$ *do not protect* (1500/0)

R06: *betw_centr* ≤ 0.004271 **and** *deg* > 6 **and** *betw_centr* ≤ 0.000121
$\Longrightarrow$ *do not protect* (159/25)

R07: *betw_centr* ≤ 0.004271 **and** *deg* > 6 **and** *deg* > 14 **and**
deg ≤ 16 **and** *betw_centr* ≤ 0.003689
$\Longrightarrow$ *do not protect* (101/16)

R08: *betw_centr* > 0.004271 **and** *deg* > 14 $\Longrightarrow$ *do not protect* (1106/479)

R09: *betw_centr* > 0.004271 **and** *deg* > 12 $\Longrightarrow$ *do not protect* (598/291)

R10: *deg* > 12 **and** *deg* > 16 $\Longrightarrow$ *do not protect* (101/27)

R11: *betw_centr* > 0.003456 **and** *deg* ≤ 12 $\Longrightarrow$ *protect* (536/255)

R12: *deg* > 6 **and** *deg* ≤ 14 $\Longrightarrow$ *do not protect* (250/80)

R13: *# the default rule* $\Longrightarrow$ *do not protect* (146/69)

Fig. 3: Rules discovered using the attributes selected by the wrapper feature selection method ($FS_{wrapper} = \{$ *deg*, *betw_centr*, *fail_min_dist* $\}$).

(as verified using the Wilcoxon test at $\alpha = 0.05$) are marked in bold. Results presented in Table 5 show, that the rules generated for $N_v = 1000$ constructed improved solutions for graphs with the number of nodes $N_v \in \{1000, 3000\}$.

3.6 Tests on Erdős-Renyi graphs

Another round of experiments was performed on graphs constructed according to the Erdős-Renyi model $G(N_v, P_{edge})$ introduced in [16]. The methodology proposed in Section 2 was followed, but the graphs used for training and testing were not REDS, but E-R graphs with $N_v = 1000$ nodes and $P_{edge} = 3.0/N_v = 0.003$. The best performing algorithm in these experiments was the algorithm using rule-based population initialization with attributes selected using $FS_{rank,4}$ method and $P_{mod/s} = 0.8$ and $P_{mod/p} = 0.6$. The median number of saved nodes from 30 runs was 343 for the rule-based population initialization and 334 when no improvement of initial populations was performed. The difference between these

Table 4: Parameters of problem instances used in the experiments aimed at testing the universality of the rules.

Test instance parameters						
N_v	R	E	S	$\overline{k}$	N_b	N_t
1000	0.100	0.15	0.500	7.35	50	200
1250	0.089	0.15	0.447	7.97	62	250
1500	0.082	0.15	0.408	8.11	75	300
1750	0.076	0.15	0.378	8.57	88	350
2000	0.070	0.15	0.350	9.16	100	400
2250	0.067	0.15	0.333	9.43	112	450
2500	0.063	0.15	0.316	10.09	125	500
3000	0.060	0.15	0.290	10.06	150	600

Table 5: Results obtained in the experiments aimed at testing the universality of the rules (median value of the best solution found, calculated from 30 runs).

N_v	None	All	Wrapper
1000	647.5	**668.0**	**680.5**
1250	629.5	**653.5**	**671.0**
1500	648.5	**666.5**	661.5
1750	627.5	643.0	628.5

N_v	None	All	Wrapper
2000	623.0	**641.0**	**651.0**
2250	656.0	665.0	664.5
2500	658.0	**670.0**	**666.0**
3000	761.5	764.0	**765.0**

results was statistically significant, as verified using the Wilcoxon test (obtained p-value: 0.0253).

4 Conclusions

In this paper a method was presented for discovering rules that allow deciding if particular nodes in a graph should be protected in the systemic risk minimization problem. The proposed method works by analyzing solutions of the problem produced by an optimization algorithm, and builds a rule set using a machine learning algorithm. In this paper the PART algorithm was used for generating the rules, because in preliminary experiments it shown promising performance. However, the proposed method is not limited to using PART, and it could also use other classifiers. Presented results suggest that better final results are obtained when the optimization algorithm starts from the initial population improved using the rule-based classifier. Because the testing was performed on a set of problem instances separate from those used for building the rules, it can be stated that the obtained rule sets have the generalization property.

Acknowledgment

This work was supported by the Polish National Science Centre under grant no. 2015/19/D/HS4/02574. Calculations have been carried out using resour-

ces provided by Wroclaw Centre for Networking and Supercomputing (http://wcss.pl), grant No. 407.

References

1. Burkholz, R., Leduc, M., Garas, A., Schweitzer, F.: Systemic risk in multiplex networks with asymmetric coupling and threshold feedback. Physica D **323-324** (2016) 64–72
2. Battiston, S., Puliga, M., Kaushik, R., Tasca, P., Caldarelli, G.: DebtRank: Too Central to Fail? Financial Networks, the FED and Systemic Risk. Scientific Reports **2** (August 2012) 541
3. Thurner, S., Poledna, S.: DebtRank-transparency: Controlling systemic risk in financial networks. Scientific Reports **3** (May 2013) 1888
4. Antonioni, A., Bullock, S., Tomassini, M.: REDS: an energy-constrained spatial social network model. In Lipson, H., Sayama, H., Rieffel, J., Risi, S., Doursat, R., eds.: ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems, MIT Press (2014)
5. Esquivel, S.C., Leiva, A., Gallard, R.H.: Multiple crossover per couple in genetic algorithms. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97). (1997) 103–106
6. Elsayed, S.M., Sarker, R.A., Essam, D.L.: Multi-operator based evolutionary algorithms for solving constrained optimization problems. Computers & Operations Research **38**(12) (2011) 1877 – 1896
7. Zhang, L., Wang, L., Zheng, D.Z.: An adaptive genetic algorithm with multiple operators for flowshop scheduling. The International Journal of Advanced Manufacturing Technology **27**(5) (Jan 2006) 580–587
8. Michalak, K.: The Sim-EA algorithm with operator autoadaptation for the multi-objective firefighter problem. In Ochoa, G., Chicano, F., eds.: Evolutionary Computation in Combinatorial Optimization. Volume 9026 of Lecture Notes in Computer Science. Springer International Publishing (2015) 184–196
9. Deb, K., Agarwal, R.: Simulated binary crossover for continuous search space. Complex Systems **9**(2) (1995) 115–148
10. Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design. Computer Science and Informatics **26** (1996) 30–45
11. S. Vijayarani, M.D.: An efficient algorithm for generating classification rules. International Journal of Computer Science and Technology (2011)
12. Machine Learning Group at the University of Waikato: Weka 3 - Data Mining Software in Java. `https://www.cs.waikato.ac.nz/ml/weka/` (2017)
13. Ott, L., Longnecker, M.: An Introduction to Statistical Methods and Data Analysis. Brooks/Cole Cengage Learning (2010)
14. Sharma, A., Dey, S.: Performance investigation of feature selection methods. CoRR **abs/1309.3949** (2013)
15. Garcia-Martinez, C., Blum, C., Rodriguez, F., Lozano, M.: The firefighter problem: Empirical results on random graphs. Computers & Operations Research (60) (2015) 55–66
16. Erdős, P.: Some remarks on the theory of graphs. Bulletin of the American Mathematical Society **53**(4) (1947) 292–294