Estimation of Distribution Algorithms for the Firefighter Problem

Krzysztof Michalak(✉)

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

Abstract. The firefighter problem is a graph-based optimization problem in which the goal is to effectively prevent the spread of a threat in a graph using a limited supply of resources. Recently, metaheuristic approaches to this problem have been proposed, including ant colony optimization and evolutionary algorithms.

In this paper Estimation of Distribution Algorithms (EDAs) are used to solve the FFP. A new EDA is proposed in this paper, based on a model that represents the relationship between the state of the graph and positions that become defended during the simulation of the fire spreading. Another method that is tested in this paper, named EH-PBIL, uses an edge histogram matrix model with the learning mechanism used in the Population-based Incremental Learning (PBIL) algorithm with some modifications introduced in order to make it work better with the FFP. Apart from these two EDAs the paper presents results obtained using two versions of the Mallows model, which is a probabilistic model often used for permutation-based problems. For comparison, results obtained on the same test instances using an Ant Colony Optimization (ACO) algorithm, an Evolutionary Algorithm (EA) and a Variable Neighbourhood Search (VNS) are presented.

The state-position model proposed in this paper works best for graphs with 1000 vertices and more, outperforming the comparison methods. For smaller graphs (with less than 1000 vertices) the VNS works best.

Keywords: Estimation of distribution algorithms · Graph-based optimization · Firefighter problem

1 Introduction

The Firefighter Problem (FFP) originally formulated by Hartnell in 1995 [10] is a combinatorial optimization problem in which spreading of fire is modelled on a graph and the goal is to protect nodes of the graph from burning. Despite the name of the problem, the same formalism can also be used to describe spreading of other threats, such as floods, diseases in humans as well as in livestock, viruses in a computer network and so on.

B. Hu and M. López-Ibáñez (Eds.): EvoCOP 2017, LNCS 10197, pp. 108–123, 2017.
DOI: 10.1007/978-3-319-55453-2_8

There are three main areas of study concerning the Firefighter Problem. The first area is the theoretical study of the properties of the problem. In the survey [7] many aspects of the FFP are discussed, for example the complexity of the problem, algorithms and special cases, such as the FFP on infinite and finite grids. Other works study the FFP on specific graph topologies, such as grids [6], trees [5], digraphs [13], etc.

Another area of study is the application of classical optimization algorithms. For example, methods such as the linear integer programming have been used for solving the single-objective version of the FFP [6]. Some authors combined linear integer programming methods with simple heuristics, such as "save vertices with highest degrees first" [9].

An area that has just recently emerged concerns attempts to employ metaheuristic methods. This line of work originated with a work by Blum et al. [2] presented at the EvoCOP conference in 2014. This first attempt was made using the Ant Colony Optimization (ACO) approach and concerned the single-objective FFP. Later the same year another paper has been published [17] in which the multiobjective version of the FFP (MOFFP) has been formulated. The next paper concerning the MOFFP employed a multipopulation evolutionary algorithm with migration [18]. The multipopulation algorithm has later been used in the non-deterministic case [20], combining simulations with evolutionary optimization in an approach known as simheuristics which was proposed in a recent survey [12] as a proper approach to nondeterministic optimization. In another paper the Variable Neighborhood Search (VNS) method has been applied to the single-objective version of the FFP [11].

The papers published so far have been based on such metaheuristic approaches as the Ant Colony Optimization (ACO) and Evolutionary Algorithms (EAs). To the best of the knowledge of the author of this paper no attempts to use EDAs for this problem have previously been made. This paper starts investigation in this direction by proposing a new State-Position (S-P) model for the use in the FFP as well as by studying other models.

The rest of this paper is structured as follows. In Sect. 2 the firefighter problem is defined. In Sect. 3 the EDA approach is described and probabilistic models tested in this paper are presented. Section 4 describes the experimental setup and presents the results. Section 5 concludes the paper.

2 Problem Definition

The Firefighter Problem is defined on an undirected graph $G = \langle V, E \rangle$ with N_v vertices. Each vertex of this graph can be in one of the states from the set $L = \{'\mathrm{B}', '\mathrm{D}', '\mathrm{U}'\}$ with the interpretation $'\mathrm{B}'$ = burning, $'\mathrm{D}'$ = defended and $'\mathrm{U}'$ = untouched. Formally, we will use a function $l : V \rightarrow L$ to assign labels to the vertices of the graph G.

Spreading of fire is simulated in discrete time steps $t = 0, 1, \ldots$. At $t = 0$, the graph is in the initial state S_0. Most commonly, in the initial state vertices from a non-empty set $\emptyset \neq S \subset V$ are burning ('B') and the remaining ones are untouched (‘U’) (no vertices are initially defended). In each time instant $t > 0$ we are allowed to assign firefighters to a predefined number N_f of still untouched (‘U’) nodes of the graph G. These nodes become defended (‘D’) and are immune to fire for the rest of the simulation. Next, fire spreads along the edges of the graph G from the nodes labelled ‘B’ to all the adjacent nodes labelled ‘U’. The simulation stops, when fire can no longer spread. It can happen either when all nodes adjacent to fire are defended (‘D’) or when all undefended nodes are burning (‘B’).

Solutions to the firefighter problem can be represented as permutations of the numbers $1, \ldots, N_v$. During the simulation, in each time step, the first N_f yet untouched nodes (‘U’) are taken from the permutation π and become defended (‘D’). In every time step exactly N_f nodes become defended, except for the final time step in which the number of the untouched nodes may be less than N_f.

The evaluation of a solution (permutation) π is performed by simulating the spread of fire from the initial state until the fire can no longer spread. During the simulation nodes of the graph G become protected in the order determined by the permutation π. In the classical version of the FFP the evaluation of the solution π is equal to the number of nodes not touched by fire (those, that are in one of the states ‘D’ or ‘U’) when the simulation stops. In the paper [17] the multiobjective version of the FFP was proposed in which there are m values $c_i(v)$, $i = 1, \ldots, m$ assigned to each node v in the graph. In the context of fire containment multiple criteria could represent, for example, the financial value $c_1(v)$ and the cultural importance $c_2(v)$ of the items stored at the node v. Multiobjective evaluation can also be useful when preventing the spread of epidemics in livestock. In such scenario we would probably be interested in protecting different species to a certain degree. In this paper a single-objective version of the FFP is studied, in order to start the work on probabilistic models for the FFP, which are naturally less complex in the single-objective case. However, to retain similarity to the multiobjective case, costs are assigned to nodes and solutions are evaluated using these costs. To stick to the formalism used for the multiobjective case, these costs will be denoted as $c_1(v)$, even though there are no $c_i(v)$ with $i > 1$ in this paper. Evaluating a solution π requires simulating how fire spreads when firefighter assignment is done according to π. The evaluation of the solution can then be calculated as the sum of the costs assigned to those nodes that are not burnt at the end of the simulation:

$$e(\pi) = \sum_{v \in V : l(v) \neq' B'} c_1(v) \tag{1}$$

where:

$c_1(v)$ - the cost assigned to the node v.

3 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) work in a way that bears a certain resemblance to Evolutionary Algorithms (EAs). The algorithm operates in a loop

Algorithm 1. A general structure of an EDA algorithm used in this paper.

```
IN: $N_{pop}$    - The size of the population
    $N_{sample}$ - The size of a sample used for probabilistic model update

$P$ := InitPopulation($N_{pop}$)
$M$ := InitModel()
$B := \emptyset$

while not StoppingCondition() do
    // Evaluation
    Evaluate(P)

    // Caching of the best specimen
    $B$ := GetBestSpecimens(P, 1)

    // Update of the probabilistic model
    $P_{sample}$ := GetBestSpecimens($P$, $N_{sample}$)
    $M$ := UpdateModel($P_{sample}$, $M$)

    // New population
    $P$ := CreateNewSpecimens($M$, $N_{pop} - 1$)
    $P := P \cup B$
```

in which a population of specimens is used to represent and evaluate solutions of a given optimization problem (see Algorithm 1). The main difference between EDAs and evolutionary algorithms is that in the EAs genetic operators are used to produce the next generation of specimens, while in EDAs a probabilistic model is built from the population and specimens for the new generation are drawn from this model. In the algorithm used in this paper the mechanism of elitism is used, that is, the best specimen is always preserved and is transferred from the previous generation to the next one.

A general structure of EDAs used in this paper is presented in Algorithm 1. This algorithm uses the following procedures:

InitPopulation - Initializes a new population by creating N_{pop} specimens, each with a genotype initialized as a random permutation of N_v elements.
InitModel - Initializes the probabilistic model. The initialization procedure depends on the chosen problem and the EDA algorithm.
StoppingCondition - Checks if the stopping condition has been satisfied. In this paper the total running time T_{max} is used as a stopping criterion.
Evaluate - Evaluates specimens in a given population. The evaluation is performed by simulating the spread of fire and, after fire no longer spreads, calculating the overall value of the nodes in the graph that are not consumed by fire.
GetBestSpecimens - Returns a given number of specimens with the highest evaluation values from a given population. This procedure is used for storing the best specimen used by the elitism mechanism and for getting a sample from the population used for updating the probabilistic model.

UpdateModel - Updates the probabilistic model based on the current population. The exact procedure depends on the chosen model and the representation of solutions.
CreateNewSpecimens - Generates a required number of specimens using the information contained in the probabilistic model.

For an EDA algorithm three elements are necessary: a probabilistic model, a method of updating this model based on a sample of specimens and a procedure for generating new solutions based on the probabilistic model. All probabilistic models used in this paper are defined on a space Π_n of permutations of a fixed length n (with $n = N_v$).

The first two, the Mallows model and the generalized Mallows model describe exponential probability distributions on the space of permutations. Even though they were proposed in 1957 [15] and 1986 [8] respectively, they are nowadays actively researched in various applications such as recommender systems [14,16]. Recently, both models have been proposed for the usage in EDAs [3,4]. Because of space limitations, the details of the Mallows models are not given in this paper. A discussion of the learning and sampling processes can be found for example in [3].

Third method studied in this paper (EH-PBIL) uses an edge histogram matrix model used, among others, in the Edge Histogram-Based Sampling Algorithm (EHBSA) [21] and the learning rule known from the PBIL algorithm [1] to update a probability matrix, which in turn is used for generating new solutions. A similar method was used in the paper [19] for the Travelling Salesman Problem (TSP), but the model update procedure used in this paper was modified to fit the specifics of the FFP.

Fourth model, named State-Position model and introduced in this paper, is dedicated for the FFP and models the relation between the state of the graph and positions at which firefighters should be placed.

3.1 The EH-PBIL Method

This method uses an edge histogram model used for example in the Edge Histogram-Based Sampling Algorithm (EHBSA) [21] - a matrix $\mathbb{P}_{[N_v \times N_v]}$ in which each element p_{ij} represents a probability that the number j will appear in good solutions right after the number i. However, the model update mechanism is different from that used in the paper [21]. Also, an additional component is added to the model which is a weight vector W_s of length N_v containing weights that are used to calculate the chance of each number $i \in \{1, \ldots, N_v\}$ being used as the first element in the permutation. This element is added because, contrary to the TSP, in the FFP it is very important which element is the first in the solution.

Learning the Edge Histogram Matrix Model

Learning of this model follows the model update rule known from the PBIL algorithm [1]. From the sample P_{sample} used for updating the EDA models the worst solution (permutation) $\pi^{(-)}$ and the best one $\pi^{(+)}$ are selected according to the evaluation function (1). From these solutions two probability matrices $\mathbb{P}^{(-)}$ and $\mathbb{P}^{(+)}$ are built. In $\mathbb{P}^{(+)}$ all the elements are set to 0 except those elements $p_{ij}^{(+)}$ for which j immediately follows i in the permutation $\pi^{(+)}$. The matrix $\mathbb{P}^{(-)}$ is built in the same way from $\pi^{(-)}$.

The matrices $\mathbb{P}^{(+)}$ and $\mathbb{P}^{(-)}$ are used to update $\mathbb{P}$ by applying the model update mechanism from PBIL for each element p_{ij} in $\mathbb{P}$ separately, except for the elements on the diagonal which are always set to 0. This process uses two learning rate parameters: the positive learning rate η_+ and the negative learning rate η_-. Their sum is denoted $\eta = \eta_+ + \eta_-$. According to the PBIL algorithm, if $p_{ij}^{(-)} = p_{ij}^{(+)}$ the element p_{ij} of $\mathbb{P}$ is set to:

$$p_{ij} = p_{ij} \cdot (1 - \eta_+) + p_{ij}^{(+)} \cdot \eta_+, \tag{2}$$

and if $p_{ij}^{(-)} \neq p_{ij}^{(+)}$ the element p_{ij} is set to:

$$p_{ij} = p_{ij} \cdot (1 - \eta) + p_{ij}^{(+)} \cdot \eta. \tag{3}$$

Finally, each element p_{ij} (except the elements on the diagonal) is mutated with probability P_{mut} by setting:

$$p_{ij} = p_{ij} \cdot (1 - \mu) + \alpha * \mu, \tag{4}$$

where:

α - a 0 or 1 value drawn randomly with equal probabilities $P(0) = P(1) = \frac{1}{2}$,
μ - a mutation-shift parameter controlling the intensity of mutation.

The weight vector W_s is modified by taking the first element k in a genotype of each specimen and increasing $W_s[k]$ by the evaluation of the specimen from which k was taken. Thus, weights for those numbers that are used as first elements in good solutions are increased.

Sampling of the Matrix Model

A new permutation π is generated from the model as follows. The first element in the permutation may be selected in two different ways. The first is to draw a number uniformly from the set $1, \ldots, N_v$. This initialization method was used in the paper [19] for the TSP, because in the TSP the solution does not depend on which element in the tour is considered the first. In the FFP it does matter, however, which node is defended as the first one. Therefore, when the model is updated, a weight vector W_s is constructed which contains weights corresponding to how often a given number was used at the first position in the permutation. The first element $\pi[1]$ is then randomly drawn from the set $\{1, \ldots, N_v\}$ with probabilities proportional to the elements of W_s. Because the second method of initialization cannot select as the first element any of the nodes that did not appear at the first position already, it turned out that it is the most effective to combine both methods of initialization. Consequently, the uniform initialization is performed with a probability P_{unif} and the weight-vector-based initialization with a probability $1 - P_{unif}$.

When generating the element $\pi[i]$, $i = 2, \ldots, N_v$ a sum $\overline{p}$ is calculated:

$$\overline{p} = \sum_{j \notin \{\pi[1],\ldots,\pi[i-1]\}} p_{\pi[i-1]j}. \tag{5}$$

If $\overline{p} > 0$ then a number $j \notin \{\pi[1], \ldots, \pi[i-1]\}$ is randomly selected with probability $\frac{p_{\pi[i-1]j}}{\overline{p}}$. If $\overline{p} = 0$, a number from $j \notin \{\pi[1], \ldots, \pi[i-1]\}$ is randomly selected with uniform probability.

3.2 The State-Position Model

The State-Position (S-P) model proposed in this paper represents the relationship between the graph state S, the position (number of the vertex) at which the firefighter was assigned v, and the mean evaluation e which was finally achieved after using this particular assignment in the graph state S. Graph states are represented as vectors of states of the vertices in the graph, so each graph state is an element of the space L^{N_v}. The model M is thus represented by a list of ordered triples:

$$M = [\langle S_1, v_1, e_1 \rangle, \langle S_2, v_2, e_2 \rangle, \ldots \langle S_n, v_n, e_n \rangle] \tag{6}$$

where $S_i \in L^{N_v}$, $v_i \in V$ and $e_i \in \mathbb{R}$ for $i = 1, \ldots, n$.

Because the same position can be defended in the same graph state, but in various solutions (attaining each the same or a different final evaluation), triples with the same elements S and v (but different e values) or multiple copies of the same triple may be constructed when analyzing solutions found in the sample P_{sample}. To reduce the size of the model, the evaluations obtained for the same state S_i and vertex v_i are averaged and only the mean e_i is stored.

Additionally, for each node v in the graph the model stores a weight $Q[v]$ calculated as the sum of the reciprocals of the positions (counting from 1) of node v in the solutions used to build the model. These weights are used for selecting nodes to defend if no selection can be done based on the M model.

Learning of the State-Position Model

The model M is built using solutions in the sample P_{sample} (see Algorithm 1). From each permutation π in the sample, several triples are generated by simulating the spreading of fire from the initial graph state. Each time a node v is protected, a pair containing the current graph state S and the node v is stored. After the simulation finishes, the final state is evaluated and triples are formed from the stored $\langle S, v \rangle$ pairs and the evaluation e. This procedure is presented in Algorithm 2. Note, that the symbol $\oplus$ used in this algorithm is an operator for adding an item to a list.

Algorithm 2. Learning of the State-Position model

```
IN:  $P_{sample}$ - A sample from the population
     $S_0$        - The initial state of the graph
     $N_f$        - The number of firefighters assigned in one time step

OUT: $M$          - The State-Position model built from the sample
     $Q$          - The vector of weights assigned to the graph nodes
                    when the model is built

$M := \emptyset$
for $\pi \in P_{sample}$ do
    // Fire spreading simulation stores graph states and the defended nodes
    $R := \emptyset$
    $S := S_0$
    while CanSpread($S$) do
        $V :=$ SelectPositions($S, \pi, N_f$)
        for $v \in V$ do
            $S[v] :=$ 'D'
            $R := R \cup \{\langle S, v \rangle\}$
        $S :=$ SpreadFire($S$)

    // Evaluation of the final graph state
    $e :=$ EvaluateState($S$)

    // Addition of the evaluated state-position pairs to the model
    for $\langle S, v \rangle \in R$ do
        $M := M \oplus \langle S, v, e \rangle$

    // Calculation of the weights of individual nodes
    for $i := 1, \ldots, N_v$ do
        $Q[\pi[i]] := Q[\pi[i]] + \frac{1}{i}$
$M :=$ CalculateMeanEvals($M$)
```

Algorithm 2 uses the following procedures:

CanSpread - Returns a logical value indicating if in the graph state S the fire can still spread, that is, if there are untouched nodes adjacent to burning ones.
SelectPositions - Returns a set of N_f numbers that correspond to untouched nodes in the state S and are placed nearest the beginning of the permutation π (i.e. the first N_f nodes in the state 'U' appearing in π).
SpreadFire - Performs one step of the fire spreading by changing to 'B' the state of all untouched ('U') nodes adjacent to the burning ('B') ones.
EvaluateState - Evaluates the final state S by calculating the sum of values assigned to those vertices that are not burning ('B') in the state S.
CalculateMeanEvals - Aggregates the evaluations stored for state-position pairs by calculating, for each unique pair $\langle S', v' \rangle$, an average evaluation from all

triples $\langle S, v, e\rangle$ in which $S' = S$ and $v' = v$. After the aggregation the model M contains triples with unique values of S and v (no duplicate pairs $\langle S, v\rangle$ exist).

After the learning of the model is completed the set M contains ordered triples $\langle S, v, e\rangle$. In each such triple a state of the graph S is combined with the position (node number) v which got protected and the average evaluation e that was finally achieved. Using this scheme, assignments that are good at a given graph state are rewarded by high final evaluations.

Sampling of the State-Position Model

Sampling of the State-Position model is performed by generating new specimens one by one using the information stored in the model. Each specimen is generated by performing the simulation of the fire spreading starting from the initial graph state S_0 and selecting positions to defend using the model M. The procedure of generating one specimen is presented in Algorithm 3.

Initially, nodes are added in a simulation loop in which the current state of the graph is compared to the states stored in the model M. The distance between graph states is measured using the Hamming distance $H(S, S')$ (which is, simply, the number of vertices $u \in 1, \ldots, N_v$ for which $S[u] \neq S'[u]$). For each triple $\langle S', v', e'\rangle$ stored in the model M such that $v' = v$ the weight $w[v]$ of the node v is increased proportionally to the evaluation e' and inversely proportionally to a function $f()$ of the Hamming distance between the current state S and the state stored in the model S'. The function $f()$ is used to determine how the weight of the position should change with the distance between graph states. In this paper the following functions were tested:

Linear - $f(x) = 1 + x$. Makes the weight of the position decrease inversely with the distance between graph states. The 1 is added to avoid errors when the states S and S' are equal and H(S, S') = 0.
Square - $f(x) = 1 + x * x$. A function whose inverse decreases faster than that of the linear one.
Sqrt - $f(x) = 1 + \sqrt{x}$. A function whose inverse decreases more slowly than that of the linear one.
Exponential - $f(x) = 3^x$. A function producing a very narrow, exponentially vanishing peak around the given graph state S. The basis of 3 was selected because there are three possible states of each node of the graph, so the value of 3^x represents the number of the graph states with all the possible values at the x positions at which S differs from S'. The exponential function produces a value of $f(x) = 1$ for $x = 0$ as the other functions, so in each case the weight assigned to the current graph state is 1.

When there is at least one node v with a positive weight $w[v]$ the selection of the node to defend is performed using a roulette wheel selection procedure with probabilities proportional to the weights in w. Otherwise, the selection is performed using weights stored in Q which are inversely proportional to the positions at which the nodes appeared in the population (only untouched nodes are considered). Thus, selection using weights from Q gives higher priorities to

Algorithm 3. Sampling of the State-Position model

```
IN:   $S_0$ - The initial state of the graph
      $N_f$ - The number of firefighters assigned in one time step
      $M$ - The State-Position model built from the sample
      $Q$ - The vector of weights assigned to the graph nodes
            when the model is built

OUT: $\pi$ - A new solution generated from the model

// Simulate spreading of fire
$\pi := \emptyset$
$S := S_0$
while CanSpread($S$) do
    $w := [0, 0, \ldots, 0]$
    $W := [0, 0, \ldots, 0]$
    for $v := 1, \ldots, N_v$ do
        if $S[v]$ = 'U' then
            for $\langle S', v', e' \rangle \in M$, s.t. $v' = v$ do
                $w[v] := w[v] + e' \frac{1}{f(H(S,S'))}$
            $W[v] := Q[v]$
    for $i := 1, \ldots, N_f$ do
        if $\sum w > 0$ then
            $v :=$ RouletteWheelSelection($w$)
        else
            $v :=$ RouletteWheelSelection($W$)
        $w[v] := 0$
        $W[v] := 0$
        $\pi := \pi \oplus v$
        $S[v] :=$ 'D'
    $S :=$ SpreadFire($S$)

// Nodes not used in the simulation are added using Q weights
$W := [0, 0, \ldots, 0]$
for $v := 1, \ldots, N_v$ do
    if $v \notin \pi$ then
        $W[v] := Q[v]$
while $\sum W > 0$ do
    $v :=$ RouletteWheelSelection($W$)
    $W[v] := 0$
    $\pi := \pi \oplus v$

return $\pi$
```

nodes that tend to appear towards the beginning of the solutions. Of course, only some nodes are defended during the simulation. The remaining ones are added to the solution π using the roulette wheel selection with probabilities proportional to weights in Q. This time nodes are used regardless of the state 'B' or 'U' in which they were during the simulation, except for the nodes in state 'D' which are already in the solution π. This last step is used in order to retain some information concerning the precedence of the nodes in the population, even if those nodes are not used for defense.

In addition to CanSpread() and SpreadFire() procedures used in Algorithm 2, Algorithm 3 uses the **RouletteWheelSelection()** procedure that performs the roulette wheel selection procedure using weights in the given vector w. An index of each of the elements in the vector can be returned with the probability proportional to the weight at that index. For example if the weight vector is $w = [12, 4, 1, 3]$ the probability of returning 1 is 0.6, for 2 it is 0.2, for 3 it is 0.05 and for 4 it is 0.15.

4 Experiments and Results

In the experiments the EDA approach was tested with four different probabilistic models (Simple Mallows, Generalized Mallows, EH-PBIL and State-Position model). In the case of the EH-PBIL model, the parameters were set following the original paper on the PBIL algorithm [1] to: $\eta_+ = 0.1$ (learning rate), $\eta_- = 0.075$ (negative learning rate), $P_{mut} = 0.02$ (mutation probability) and $\mu = 0.05$ (mutation shift parameter). The values of $P_{unif} = 0.0$, 0.2, 0.4, 0.6, 0.8 and 1.0 were tested in order to determine the influence of this parameter introduced in this paper on the working of the algorithm. The State-Position model was used with the four functions mentioned before (Exponential, Linear, Sqrt and Square).

For comparison, tests were performed with the Ant Colony Optimization (ACO) algorithm proposed in [2] and the VNS method proposed in [11]. An Evolutionary Algorithm (EA) was also tested with three crossover operators that performed best in the previous paper [17], that is the CX, OBX and PBX. For the mutation operator the insertion mutation was used because it worked best in the aforementioned paper. Crossover and mutation probabilities were set to $P_{cross} = 0.9$ and $P_{mut} = 0.05$. The population size was set to $N_{pop} = 100$ for all the methods. The sample size for the EDAs was set to 20% of the population size, so $N_{sample} = 20$. The EDAs and EAs employed the elitism mechanism in which one, the best, solution was always promoted to the next generation.

For testing Erdős-Renyi graphs, represented using adjacency matrices, with $N_v = 500$, 750, 1000, 1250, 1500, 1750, 2000, 2250, 2500 and 5000 vertices were used. The probability with which an edge was added between any two different vertices (independently of the others) was $P_{edge} = \frac{3}{N_v}$. This value was selected during preliminary experiments in such a way that the obtained problem instances were not too easy (the entire graph easily protected) nor too difficult (all the nodes except the defended ones always lost). The other parameters of the problem instances were $N_s = 1$ starting point and $N_f = 2$ firefighters allowed per a time step. In order to ensure that the generated instances were difficult enough, only such graphs were used in which the number of edges adjacent to the starting points exceeded the number N_f of firefighters allowed per a time step. This requirement was formulated to eliminate a trivial solution which is to use the N_f firefighters to cut off the starting points from the rest of the graph during the first time step. While such a solution is a very good one (most of the graph is saved in such case) it is also trivial to apply and therefore is not really indicative of the actual problem solving capacity of the tested algorithms. Costs drawn uniformly from the range $[0, 100]$ were assigned to the nodes of the graphs. A set of 50 different graphs was prepared as described above for each graph size N_v and each method was tested on the same 50 graphs.

Comparison of the algorithms was done on the basis of the median calculated over these 50 runs from the evaluations of the best solution found by each of the algorithms. Median values were used because statistical testing could then be performed using the Wilcoxon test without ensuring normality of the distributions. The results were compared at T_{max} = 300, 600, 900 and 7200 s for N_v = 500, 750, 1000; N_v = 1250, 1500, 1750; N_v = 2000, 2250, 2500; and N_v = 5000 respectively (see Tables 1 and 2). In the tables the best value for each test problem size N_v is marked in bold.

Table 1. Median value of the saved nodes obtained in the experiments.

N_v		500	750	1000	1250	1500
T_{max}		300 s			600 s	
ACO		$1.369 \cdot 10^3$	$1.332 \cdot 10^3$	$1.320 \cdot 10^3$	$3.331 \cdot 10^3$	$3.837 \cdot 10^3$
EA	CX	$1.834 \cdot 10^3$	$1.831 \cdot 10^3$	$1.909 \cdot 10^3$	$4.023 \cdot 10^3$	$4.614 \cdot 10^3$
	OBX	$2.039 \cdot 10^3$	$2.035 \cdot 10^3$	$2.027 \cdot 10^3$	$3.993 \cdot 10^3$	$4.504 \cdot 10^3$
	PBX	$2.013 \cdot 10^3$	$2.057 \cdot 10^3$	$2.013 \cdot 10^3$	$3.971 \cdot 10^3$	$4.474 \cdot 10^3$
Mallows	Generalized	$1.421 \cdot 10^3$	$1.337 \cdot 10^3$	$1.388 \cdot 10^3$	$3.757 \cdot 10^3$	$4.305 \cdot 10^3$
	Simple	$1.523 \cdot 10^3$	$1.525 \cdot 10^3$	$1.495 \cdot 10^3$	$3.790 \cdot 10^3$	$4.398 \cdot 10^3$
EH-PBIL	0.0	$1.470 \cdot 10^3$	$1.442 \cdot 10^3$	$1.465 \cdot 10^3$	$3.697 \cdot 10^3$	$4.249 \cdot 10^3$
	0.2	$1.450 \cdot 10^3$	$1.482 \cdot 10^3$	$1.502 \cdot 10^3$	$3.743 \cdot 10^3$	$4.276 \cdot 10^3$
	0.4	$1.454 \cdot 10^3$	$1.471 \cdot 10^3$	$1.450 \cdot 10^3$	$3.780 \cdot 10^3$	$4.314 \cdot 10^3$
	0.6	$1.386 \cdot 10^3$	$1.420 \cdot 10^3$	$1.413 \cdot 10^3$	$3.703 \cdot 10^3$	$4.261 \cdot 10^3$
	0.8	$1.364 \cdot 10^3$	$1.350 \cdot 10^3$	$1.361 \cdot 10^3$	$3.634 \cdot 10^3$	$4.174 \cdot 10^3$
	1.0	$1.296 \cdot 10^3$	$1.343 \cdot 10^3$	$1.344 \cdot 10^3$	$3.646 \cdot 10^3$	$4.138 \cdot 10^3$
State-Position	Exponential	$1.821 \cdot 10^3$	$1.875 \cdot 10^3$	$1.951 \cdot 10^3$	$4.229 \cdot 10^3$	$4.831 \cdot 10^3$
	Linear	$1.942 \cdot 10^3$	$2.057 \cdot 10^3$	$2.195 \cdot 10^3$	$4.464 \cdot 10^3$	$5.025 \cdot 10^3$
	Sqrt.	$1.694 \cdot 10^3$	$1.812 \cdot 10^3$	$1.955 \cdot 10^3$	$4.173 \cdot 10^3$	$4.715 \cdot 10^3$
	Square	$2.076 \cdot 10^3$	$2.178 \cdot 10^3$	$\mathbf{2.372 \cdot 10^3}$	$\mathbf{4.643 \cdot 10^3}$	$\mathbf{5.216 \cdot 10^3}$
VNS		$\mathbf{2.832 \cdot 10^3}$	$\mathbf{2.678 \cdot 10^3}$	$2.351 \cdot 10^3$	$4.521 \cdot 10^3$	$5.121 \cdot 10^3$
FWER		$2.09 \cdot 10^{-8}$	$5.53 \cdot 10^{-7}$	$8.43 \cdot 10^{-1}$	$1.04 \cdot 10^{-2}$	$5.20 \cdot 10^{-4}$

Table 2. Median value of the saved nodes obtained in the experiments.

N_v		1750	2000	2250	2500	5000
T_{max}		600 s	900 s			7200 s
ACO		$4.314 \cdot 10^3$	$1.425 \cdot 10^3$	$5.207 \cdot 10^3$	$5.828 \cdot 10^3$	$1.525 \cdot 10^3$
EA	CX	$4.874 \cdot 10^3$	$2.148 \cdot 10^3$	$5.811 \cdot 10^3$	$6.338 \cdot 10^3$	$2.454 \cdot 10^3$
	OBX	$4.796 \cdot 10^3$	$2.170 \cdot 10^3$	$5.863 \cdot 10^3$	$6.339 \cdot 10^3$	$2.476 \cdot 10^3$
	PBX	$4.808 \cdot 10^3$	$2.139 \cdot 10^3$	$5.777 \cdot 10^3$	$6.348 \cdot 10^3$	$2.408 \cdot 10^3$
Mal-lows	Generalized	$4.674 \cdot 10^3$	$1.606 \cdot 10^3$	$5.494 \cdot 10^3$	$6.205 \cdot 10^3$	$1.746 \cdot 10^3$
	Simple	$4.736 \cdot 10^3$	$1.620 \cdot 10^3$	$5.631 \cdot 10^3$	$6.257 \cdot 10^3$	$1.803 \cdot 10^3$
EH-PBIL	0.0	$4.678 \cdot 10^3$	$1.546 \cdot 10^3$	$5.626 \cdot 10^3$	$6.238 \cdot 10^3$	$1.749 \cdot 10^3$
	0.2	$4.709 \cdot 10^3$	$1.652 \cdot 10^3$	$5.617 \cdot 10^3$	$6.238 \cdot 10^3$	$1.781 \cdot 10^3$
	0.4	$4.712 \cdot 10^3$	$1.610 \cdot 10^3$	$5.505 \cdot 10^3$	$6.239 \cdot 10^3$	$1.768 \cdot 10^3$
	0.6	$4.732 \cdot 10^3$	$1.566 \cdot 10^3$	$5.479 \cdot 10^3$	$6.218 \cdot 10^3$	$1.712 \cdot 10^3$
	0.8	$4.649 \cdot 10^3$	$1.499 \cdot 10^3$	$5.471 \cdot 10^3$	$6.189 \cdot 10^3$	$1.673 \cdot 10^3$
	1.0	$4.685 \cdot 10^3$	$1.493 \cdot 10^3$	$5.490 \cdot 10^3$	$6.152 \cdot 10^3$	$1.627 \cdot 10^3$
State-Position	Exponential	$5.287 \cdot 10^3$	$2.240 \cdot 10^3$	$6.081 \cdot 10^3$	$6.858 \cdot 10^3$	$2.476 \cdot 10^3$
	Linear	$5.527 \cdot 10^3$	$2.534 \cdot 10^3$	$6.445 \cdot 10^3$	$7.059 \cdot 10^3$	$2.982 \cdot 10^3$
	Sqrt.	$5.200 \cdot 10^3$	$2.224 \cdot 10^3$	$6.161 \cdot 10^3$	$6.703 \cdot 10^3$	$2.640 \cdot 10^3$
	Square	$\mathbf{5.713 \cdot 10^3}$	$\mathbf{2.756 \cdot 10^3}$	$\mathbf{6.591 \cdot 10^3}$	$\mathbf{7.252 \cdot 10^3}$	$\mathbf{3.209 \cdot 10^3}$
VNS		$5.253 \cdot 10^3$	$2.019 \cdot 10^3$	$6.092 \cdot 10^3$	$6.514 \cdot 10^3$	$1.647 \cdot 10^3$
FWER		$2.63 \cdot 10^{-7}$	$1.04 \cdot 10^{-2}$	$1.36 \cdot 10^{-8}$	$1.55 \cdot 10^{-8}$	$2.131 \cdot 10^{-7}$

The comparison with respect to the running time of the algorithms was chosen because of a large variety of methods used for comparison (the same approach was used in the paper on VNS [11]). Two other commonly used comparison criteria, the number of generations and the number of solution evaluations, are not well-suited for the comparisons made in this paper. Comparison by the number of generations overlooks the fact that certain methods may perform additional, costly computations in each generation, which is not uncommon in the EDAs. Comparison by the number of solution evaluations suffers from the same problem and also may not be reliable in the case of different solution representations (e.g. those used by the EA/EDA and the VNS).

From the results presented in the tables it can be seen, that for smaller graphs the VNS is very effective, but for larger graphs the State-Position EDA performs best. The tables contain also the value of the Family-Wise Error Rate for the hypothesis that the median values produced by the best performing method for a given N_v are statistically different than those produced by the other methods. This FWER value was calculated as $1 - \prod_i (1 - v_i)$, where v_i are p-values obtained in pairwise comparisons between the best performing method and each of the other methods. The calculated value is the upper bound for the probability that at least one of the comparison methods attains the same median values as the best-performing one. For graph sizes of $N_v = 1250$ and more the calculated FWER values are at most $1.04 \cdot 10^{-2}$. This shows that the difference between the State-Position EDA with the Square function and the other methods is statistically significant. For $N_v = 1000$ the State-Position EDA still produced the best result, but because of the small difference from the results produced by the VNS the FWER is high, so the statistical significance cannot be confirmed. For $N_v = 500$ and 750 the State-Position EDA was outperformed by the VNS, with the difference statistically significant in both cases.

The results produced by the algorithms for $N_v = 2500$ with respect to the running time are shown in Fig. 1. Note, that in case when multiple variants or parametrizations of one method were tested, only one variant is presented in the figure, the one that produced the best result at $T_{max} = 900$ s. The algorithms presented in the figure are the ACO, the EA using the OBX crossover, the Simple Mallows, the EH-PBIL with $P_{unif} = 0.4$, the State-Position EDA with the Square function and the VNS.

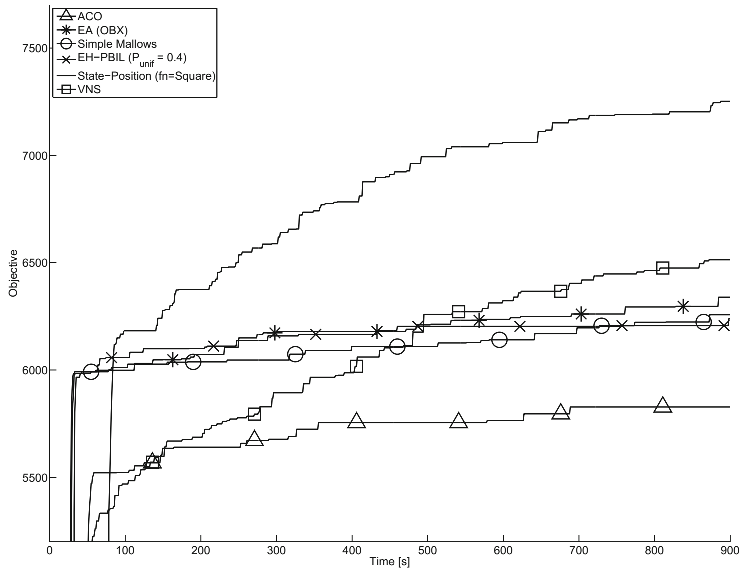

Fig. 1. Results produced by the algorithms for $N_v = 2500$ with respect to the running time. Note, that in case when multiple variants or parametrizations of one method were tested, only one variant is presented in the figure, the one that produced the best result at $T_{max} = 900$ s.

5 Conclusion

In this paper EDA algorithms using several probabilistic models were tested for the Firefighter Problem (FFP). A new model was proposed which represents the relationship between the state of the graph and the positions which are defended by firefighters. The proposed State-Position model outperformed the other EDAs, in particular the Mallows model which is commonly used for permutation-based problems and therefore was used in this paper as one of the comparison models. Compared to all the tested methods, the State-Position EDA produced the best results for graphs with $N_v = 1000$ and more. For $N_v = 500$ and 750 the VNS method shown the best performance.

The results presented in this paper form an interesting starting point for further research. First, the State-Position model builds a representation that is easily interpretable in the domain of the problem. This may constitute a basis for knowledge extraction, for example in the form of rules guiding the placement of firefighters in various graph states. Another possible extension is to apply EDAs to the multiobjective FFP either by developing multiobjective models, or by using the Sim-EDA approach proposed in [19].

Acknowledgment. Calculations have been carried out using resources provided by Wroclaw Centre for Networking and Supercomputing (http://wcss.pl), grant No. 407.

References

1. Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Carnegie Mellon University, Pittsburgh, PA, USA, Technical report (1994)
2. Blum, C., Blesa, M.J., García-Martínez, C., Rodríguez, F.J., Lozano, M.: The firefighter problem: application of hybrid ant colony optimization algorithms. In: Blum, C., Ochoa, G. (eds.) EvoCOP 2014. LNCS, vol. 8600, pp. 218–229. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44320-0_19
3. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. IEEE Trans. Evol. Comput. **18**(2), 286–300 (2014)
4. Ceberio, J., Mendiburu, A., Lozano, J.A.: Introducing the mallows model on estimation of distribution algorithms. In: Lu, B.-L., Zhang, L., Kwok, J. (eds.) ICONIP 2011. LNCS, vol. 7063, pp. 461–470. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24958-7_54
5. Costa, V., Dantas, S., Dourado, M.C., Penso, L., Rautenbach, D.: More fires and more firefighters. Discrete Appl. Math. **161**(16–17), 2410–2419 (2013)
6. Develin, M., Hartke, S.G.: Fire containment in grids of dimension three and higher. Discrete Appl. Math. **155**(17), 2257–2268 (2007)
7. Finbow, S., MacGillivray, G.: The firefighter problem: a survey of results, directions and questions. Australas. J. Comb. **43**, 57–77 (2009)
8. Fligner, M.A., Verducci, J.S.: Distance based ranking models. J. R. Stat. Soc. **48**(3), 359–369 (1986)
9. Garca-Martnez, C., et al.: The firefighter problem: empirical results on random graphs. Comput. Oper. Res. **60**, 55–66 (2015)
10. Hartnell, B.: Firefighter! An application of domination. In: 20th Conference on Numerical Mathematics and Computing (1995)
11. Hu, B., Windbichler, A., Raidl, G.R.: A new solution representation for the firefighter problem. In: Ochoa, G., Chicano, F. (eds.) EvoCOP 2015. LNCS, vol. 9026, pp. 25–35. Springer, Heidelberg (2015). doi:10.1007/978-3-319-16468-7_3

12. Juan, A.A., et al.: A review of simheuristics: extending metaheuristics to deal with stochastic combinatorial optimization problems. Oper. Res. Perspect. **2**, 62–72 (2015)
13. Kong, J., Zhang, L., Wang, W.: The surviving rate of digraphs. Discrete Math. **334**, 13–19 (2014)
14. Lu, T., Boutilier, C.: Learning mallows models with pairwise preferences. In: Getoor, L., Scheffer, T. (eds.) Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 145–152. ACM (2011)
15. Mallows, C.L.: Non-null ranking models. Biometrika **44**(1–2), 114–130 (1957)
16. Meila, M., Chen, H.: Dirichlet process mixtures of generalized mallows models. Uncertainty Artif. Intell. **1**(1), 358–367 (2010)
17. Michalak, K.: Auto-adaptation of genetic operators for multi-objective optimization in the firefighter problem. In: Corchado, E., Lozano, J.A., Quintián, H., Yin, H. (eds.) IDEAL 2014. LNCS, vol. 8669, pp. 484–491. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10840-7_58
18. Michalak, K.: The Sim-EA algorithm with operator autoadaptation for the multiobjective firefighter problem. In: Ochoa, G., Chicano, F. (eds.) EvoCOP 2015. LNCS, vol. 9026, pp. 184–196. Springer, Heidelberg (2015). doi:10.1007/978-3-319-16468-7_16
19. Michalak, K.: Sim-EDA: a multipopulation estimation of distribution algorithm based on problem similarity. In: Chicano, F., Hu, B., García-Sánchez, P. (eds.) EvoCOP 2016. LNCS, vol. 9595, pp. 235–250. Springer, Heidelberg (2016). doi:10.1007/978-3-319-30698-8_16
20. Michalak, K., Knowles, J.D.: Simheuristics for the multiobjective nondeterministic firefighter problem in a time-constrained setting. In: Squillero, G., Burelli, P. (eds.) EvoApplications 2016. LNCS, vol. 9598, pp. 248–265. Springer, Heidelberg (2016). doi:10.1007/978-3-319-31153-1_17
21. Tsutsui, S.: Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. In: Guervós, J.J.M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., Fernández-Villacañas, J.-L. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 224–233. Springer, Heidelberg (2002). doi:10.1007/3-540-45712-7_22