PATH FINDING IN A TERRAIN WITH OBSTACLES: AN EXPERIMENTATION SYSTEM AND ALGORITHMS COMPARISON

Lukasz Gadek, Leszek Koszalka, Iwona Pozniak-Koszalka, Andrzej Kasprzak
Department of Systems and Computer Networks, Wroclaw University of Technology,
Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland
E-mail: leszek.koszalka@pwr.wroc.pl

ABSTRACT
In this paper, the research on a two dimensional path problem is conducted using a designed and implemented experimentation system. Studies are performed with respect to various parameters and distinct path problem categories: shortest path, pseudo Hamilton cycle, decisive problem. Two heuristic algorithms for solving such problems are created, including the Algorithm Based on Simulated Annealing (ABSA) and the Algorithm Based on Ant Colony (ABAC). The properties of algorithms are discussed with reference to Bellman - Ford deterministic algorithm. Analysis of results of the designed simulation experiments allows recommending the both proposed algorithms.

KEYWORDS
Path problem, optimization, algorithm, simulated annealing, ant colony, experimentation system.

1. Introduction
Optimization tasks are the most common problems being solved by modern science. With an appropriate definition of a goal function and feasible solution space, many technological tasks can be reduced to the classical optimization. Related issues are: optimal control [1], [2], telecommunication [3], robotics [4], [5], resource allocation [6], [7] and many more. Description of problem must fulfil one constriction - the goal function must be precisely known, due to being a fundamental source of information for optimization algorithm.

Usually, the two types of algorithms are taken into consideration: heuristic and deterministic - referred also as specialised method due to extensive a priori knowledge requirement. The main difference in an approach is the form of output. In deterministic - it is always the same constant for the same input parameters, while for heuristic – it differs, i.e., output could be treated as a stochastic process.

Even if average result of heuristic is away from the global optimum (as in case of the set 3 in Fig. 1), it still can be reached with N-th re-run of algorithm (where N is usually a large number). To increase the chance of reaching optimal solution, range of obtained results should be shifted or extended in the direction of optimum by respectively: lowering average or increasing variance of a resulting set. In Fig. 1, the set (denoted by 1) has the biggest variance but its mean is too high, so an obtained range of solutions is unsatisfactory. The other set (denoted by 2) has sufficient mean but its variance is too small to cover global minimum. Only the set (denoted by 3) reaches an optimal solution.

Figure 1. Example of distribution of heuristic results.

In the paper, a comparison of results obtained with the newly created (by the authors of this paper) algorithms ABSA and ABAC, and results obtained using implementation of a Bellman – Ford concept modified to be used with various path problems, is based on the average and variance of resulting sets.

The rest of the paper is organized as follows. In Section 2, the related works are shortly presented. Section 3 contains problem statement and short description of experimentation system. In Section 4, the created algorithms are described. Section 5 contains some detailed results of investigations and a brief analysis of results of series of simulation experiments made for three categories of problem. The final remarks appear in Section 6.

2. Related Works
As modern optimization problems are becoming more complex, the interest in heuristics is increasing. The effect could be seen in numerous papers on subject of the analysis and possible applications of those.

The majority of studies are being focused on a specific path problem. The papers include simple terrain structure cases, e.g., the shortest path problem (SPP) in a node-type structure [8] and a modified Hamiltonian cycle problem [9]. In this paper, the approach and problem
definition are flexible due to the innovative experimentation system.

The studies with more generalised approach, e.g. modifiable 2D mesh [10] and practical time sensitive application [11] (path finding in computer games), are rare. Results and conclusion in those are often analysed with respect to the researched case only.

To avoid ‘narrow spectrum’ approach, considerations from [12] are adapted. This is being followed by stochastic analysis of resulting sets (as in Fig. 1) and the detailed studies on ‘input parameters – output results’ relationship.

3. Path Problem

3.1 Problem Statement

In general, path problems concerns finding a route that can optimize a goal function. Most often, feasible paths (also referred as an area or terrain) can be represented by graph structure. In the paper, it is considered the two-dimensional case where the set of feasible paths is represented as a mesh in which every vertex is linked with other four vertices except of border edges (see Fig. 2).

![Example of mesh and its representation](image)

The path $R$ can be described as $n$-elements vector (1) of dynamic length. Vector $R$ consists of ordered pairs of integer values (2), which describe nodes (vertices) in mesh belonging to the path in respective order, including ‘start’ node (3) and ‘end’ node (4).

$$R = [R_1, R_2, R_3, \ldots, R_n]$$  
$$R_i = (x_i, y_i)$$  
$$R_1 = (x_{start}, y_{start})$$  
$$R_n = (x_{stop}, y_{stop})$$  
$$R_{i+1} = (x_{i} \pm 1, y_{i}) \lor (x_{i}, y_{i} \pm 1)$$  
$$R_i \neq R_j \forall (i \neq j)$$

The route must be continuous over all $n$ – elements (5) and may not intersect (6). Basically, the task is to find vector $R$ satisfying all of constraints, including (3) - (6), such that the value of the goal function defined by (9) is the closest to the optimal value, i.e., $F(R) \approx F_{optimal}$.

3.2 Experimentation System

The designed and implemented experimentation system, called AreaNavigator, allows creating 2D path problem and evaluating predefined or embedded algorithms. The software is written in Java and uses the SQL database.

The AreaNavigator contains three modules:

- **DB module** – database that stores created meshes (scenarios) and the obtained results,
- **Visualization module** - GUI software, which can show areas and routes found by the tested algorithms,
- **Algorithms module** – programs in Java for performing the considered algorithms. There is also available handling mechanism for adding new algorithms.

The innovative aspect of the AreaNavigator is due to introduction of various types of vertices (field types), interpreted as obstacles and checkpoints. The system allows to create a mesh consisting of $W \times H$ rectangular terrain containing vertices interpreted as obstacles, including tree (t), rock (r), windmill (w) and vertices being checkpoints, including house (h), start, stop, or vertices being empty (e). Crossing obstacles or checkpoints (including these nodes into vector $R$) corresponds to penalties or rewards, respectively. To every node (the pair of coordinates $(x, y)$) is assigned the cost corresponding to one of seven costs (7), denoted by $P$ with proper index, and determined by the user of the experimentation system.

$$A(x, y) \in \{P_{start}, P_{stop}, P_h, P_t, P_r, P_w\}$$  

(7)

The AreaNavigator is designed to solving minimization problem: $\min F(R)$, where the goal function $F(R)$ is defined as the sum of penalties (8).

$$F(R) = \sum_{i=1}^{n} A(R_i)$$  

(8)

As stated in (7) and (8) every field type has its own cost value $P_{fieldtype}$. It is assumed that (i) ‘start’ node and ‘end’ node do not affect cost function ($P_{start} = P_{stop} = 0$), (ii) every house produces reward $P_h$ which is interpreted as decreasing the overall penalty corresponding with not visiting all houses, i.e., visiting all houses means no penalty connected to field types of this kind. This ‘mechanism’ ensures that always $F(R) \geq 0$. Taking into account these assumptions, the goal function (8) can be expressed by the formula (9).

$$F(R) = (Q_h - C_h)P_h + C_tP_t + C_rP_r + C_wP_w$$  

(9)

where:

$Q_h$ – the total number of houses on a given terrain,
C_{field name} – the number of intersections of a path R and specific field type (determined by an index of C).

An example of a path and calculation of \( F(R) \) made along with (9) is shown in Fig. 3.

\[
R = [(1,1), (2,1), (2,2), (2,3), (2,4), (3,4)]
\]

\[
A(R) = [P_{\text{start}}; P_1; P_2; P_3; P_4; P_{\text{stop}}]
\]

\[
F(R) = 2P - 1P + 1P + 1P + 0P + 1P
\]

\[
= 3P + P + P + P
\]

Fig. 3 presents a path \( R \) defined by (1). This path can be considered as a solution for the terrain 5x5 with two houses and 15 obstacles, including 8 trees, 5 rocks and 2 windmills located in some nodes. The ‘end’ node is denoted as ‘stop’.

Many of real life problems can be simulated using the AreaNavigator, e.g., planning a railroad route, in which the goal is to include as many cities as possible while avoiding rough terrain. For this example, the cost (penalties) associated with vertices located in a forest should be greater than for those in a plain field.

3.3 Categories of Path Problem

The overall path problem can be divided into large quantity of sub-problems. The diversity of a problem definition can be reached with:
(i) Adjusting cost values of specific obstacles and checkpoints,
(ii) Modification of obstacles displacement in the terrain,
(iii) Changing density of objects presence in the area.

The designed experimentation system gives opportunities for multi-aspects evaluation of properties of heuristic algorithms for solving path problems and their comparison. Thus far, majority of such comparisons have been performed on the basis of one type of problem or on finite and small variety of problems (see Section 2).

Due to flexible goal function statement (9), the variety of problems can be easily extended (with adjustable penalties and rewards values). Despite to an almost infinite number of combinations, this paper is focusing on three main categories of path problems:
- **Pseudo Hamilton cycles** - connected to Traveling Salesman Problem (with house reward relatively high),
- **Shortest Path Problem** (with house reward negligibly low),
- **Decisive problem** (with both penalties and rewards on comparable level).

The difference between these problems is illustrated on example in Fig. 4, where the near-to-optimal solutions (acquired with Bellman-Ford algorithm) are shown.

![Bellman-Ford solutions for three types of problem.](image)

The AreaNavigator allows conducting experiments with any other category (type of path problem, which can be described following formulas (1) – (9)).

4. Algorithms

4.1 Algorithm Based on Simulated Annealing (ABSA)

The concept of ABSA consists in some modifications of Simulated Annealing method [13]. The introduced improvements consist in implementing anti-lock mechanism and temperature dependent randomness. These improvements can be seen in the ABSA step-by-step description provided below. The number of iterations of the algorithm is controlled with population and temperature related input parameters.

**Input parameters:**
\( T_{\text{start}} \) temperature at start,  
\( T_{\text{stop}} \) temperature at end (stop condition),  
\( \alpha \) temperature drop with new epoch; \( \alpha \in (0,1) \),  
\( \text{pop} \) population (number of cut-off operations executed in every epoch),  
\( K \) acceptance coefficient – used for adjustment of \( f_{\text{acceptance}} \) (step 5 of ABSA).

**ABSA**

1: generate randomly first (initial) population – choose the best path;
2: decrease temperature \( T_i = \alpha T_{i-1} \); if \( T_i < T_{\text{stop}} \) then stop algorithm - present \( R_{\text{best}} \) is an output;
3: cut off a random part of the path (length of cut depending on \( T_i / T_{\text{start}} \)) and randomly fill the gap;
4: check whether filled up path is better than original, if yes then \( R_{\text{best}} = R_i \) and go to 6;
5: generate at random \( r \in (0,1) \); if \( r < \exp\left\{ F(R_{\text{best}}) - F(R_i) \right\} / (KT) \), i.e. \( r < f_{\text{acceptance}}(K, T, \Delta F(R)) \), then \( R_{\text{best}}=R_i \);
6: if whole population (\( \text{pop} \)) of epoch is not yet generated go to 3, else go to 2 and \( i+i \).
The crucial part (step 3) of the algorithm, which is responsible for route improvement, is illustrated on an example in Fig. 5.

Figure 5. Example of a ‘cut off and fill mechanism’ in ABSA.

The anti-lock mechanism (step 5 of ABSA - $f_{\text{acceptance}}$) is dependent mostly on $K$, while the iteration quantity is determined by $\alpha$ and $T_{\text{start}}$ to $T_{\text{stop}}$ ratio. The randomness of a cut-off gap generation is strongly connected to temperature drop ratio $\alpha$. Hence, three strategies of searching with ABSA can be distinguished: local, global and precise. The discrepancies in result are shown in Fig. 6.

Local search with low $\alpha$ and high difference between $T_{\text{start}}$ and $T_{\text{stop}}$ is illustrated in the left picture of Fig. 6. The best path found in the first epoch is modified, so it reaches optimal route in neighbourhood of a first generated path.

Global search principle is opposite - $\alpha$ is close to 1 and difference between $T_{\text{start}}$ and $T_{\text{stop}}$ is small. It cuts-off extended path sections. Thus, it searches for optimum from global perspective, with less efficiency of optimization in closest neighbourhood, as it is shown on picture in the middle of Fig. 6.

Precise search is the slowest but also the most accurate strategy. It performs the largest number of iterations, due to high $\alpha$ and high difference between $T_{\text{start}}$ and $T_{\text{stop}}$. It is combination of both previous types as it is shown in the right picture of Fig. 6. In a few first epochs, it performs a global search and afterwards smoothly proceeds to a local search.

Another crucial factor is the acceptance coefficient $K$. It defines the probability of replacing the best path with actual one despite the goal function relation (step 5 of the algorithm). Appropriate adjustment of $K$ has a major influence on the output variance as it is controlling the anti-lock mechanism (resistance to a local minimum trap). The noticeable displacement of best results can be observed due to the various $K$ used (Table 1). Thus, one of the main difficulties in applying ABSA is appropriate adjustment of $K$, which must be empirically based - there are no rules available to do it with a priori knowledge - $K$ is selected via comparison of results of series of experiments.

<table>
<thead>
<tr>
<th>K</th>
<th>average</th>
<th>std dev</th>
<th>best</th>
<th>average</th>
<th>std dev</th>
<th>best</th>
<th>average</th>
<th>std dev</th>
<th>best</th>
<th>average</th>
<th>std dev</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-1}$</td>
<td>18279</td>
<td>989.78</td>
<td>16600</td>
<td>12733</td>
<td>524.80</td>
<td>12000</td>
<td>20160</td>
<td>3115.09</td>
<td>15700</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>17850</td>
<td>903.63</td>
<td>16100</td>
<td>12540</td>
<td>560.54</td>
<td>11800</td>
<td>19800</td>
<td>2864.23</td>
<td>15500</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>17935</td>
<td>938.98</td>
<td>16200</td>
<td>12446</td>
<td>383.20</td>
<td>11800</td>
<td>19776</td>
<td>2834.12</td>
<td>15400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>17756</td>
<td>974.20</td>
<td>15800</td>
<td>12546</td>
<td>499.28</td>
<td>11800</td>
<td>19735</td>
<td>2678.71</td>
<td>15600</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>17905</td>
<td>929.56</td>
<td>16700</td>
<td>12490</td>
<td>385.88</td>
<td>11800</td>
<td>19463</td>
<td>2566.31</td>
<td>15100</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.2 Algorithm Based on Ant Colony (ABAC)

ABAC is implemented basing on ideas described in [14]. It is an example of swarm algorithm, which operates on large population of related particles (ants / paths). ABAC main mechanism (shown in step by step description provided below) is based on a pheromone. Ants are producing the pheromone trace on their paths (with intensity depending on $F(R_{max})$ reached), the next generation individuals are making their ways in pseudo-random order using the trace left by past generations.

Input parameters:
- $gen$ number of generations (stop condition),
- $pop$ quantity of ants (generated paths) in a generation,
- $maxP$ maximum production of the pheromone for one ant per field,
- $\alpha$ pheromone evaporation rate.

ABAC

1: set the same pheromone values on a whole area (except houses, where it is doubled); $i=1$;
2: generate the population of $pop$ paths influenced by a pheromone using (10) – see Fig. 7;
3: calculate the goal function of obtained routes; if $F(R_{head}) > F(R_j)$ then $R_{best} = R_j$ (where $j$ is the number of ant in epoch $t$-th);
4: increase the pheromone of fields passed by every ant of generation with $\Delta Phr = (F(R_i)/maxP)/(F(R_{best})$;
5: decrease all pheromone $Phr_{i+1} = \alpha Phr$;
6: if $i < gen$ then $i++$ and go to step 2 else stop.

$$\text{rand}(r) = \left( 0, \sum_k [Phr(x+k,y) + Phr(x,y+k)] \right)$$

(10)

Coefficients $pop$ and $gen$ are affecting the number of iterations, which has a direct influence on a computational time and an obtained result. The step 2 of the algorithm is illustrated in Fig. 7.

Table 2. ABAC output in respect to $C$ coefficient with constant $maxP = 1$. (the best results are blackened).

<table>
<thead>
<tr>
<th>Decisive problem: $P_c=100$, $P_r=200$, $P_o=200$, $P_m=2000$.</th>
<th>SPP: $P_c=100$, $P_r=200$, $P_o=300$, $P_m=400$, $P_b=800$.</th>
<th>Pseudo Hamilton cycle: $P_c=100$, $P_r=200$, $P_o=300$, $P_m=400$, $P_b=30000$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>$\alpha$</td>
<td>avg</td>
</tr>
<tr>
<td>40</td>
<td>0.1</td>
<td>20845</td>
</tr>
<tr>
<td>50</td>
<td>0.5</td>
<td>20230</td>
</tr>
<tr>
<td>120</td>
<td>0.3</td>
<td>19647</td>
</tr>
<tr>
<td>200</td>
<td>0.5</td>
<td>19096</td>
</tr>
<tr>
<td>280</td>
<td>0.7</td>
<td>18366</td>
</tr>
<tr>
<td>350</td>
<td>0.5</td>
<td>18701</td>
</tr>
<tr>
<td>360</td>
<td>0.9</td>
<td>16945</td>
</tr>
</tbody>
</table>

Figure 5. The generation of direction of ant in ABAC (step 2) in dependence on random $r$ and pheromone values.

The goal function heads exponentially towards the optimum with the increase of iterations. However, the most significant describing value (due to algorithm result) was empirically defined as a coefficient $C$ which is proportional to three input parameters: the quantity of ants and pheromone related parameters as in (11).

$$C = maxP \alpha pop$$

(11)

The coefficient $C$ has a major impact on acquired output. If it is too high then search mechanism will not be sufficiently random due to the high pheromone trace values. Thus, the resemblance between final results and those obtained in early epochs can be observed. This phenomenon has negative influence on the obtained set of results that are characterized usually by unsatisfactory average.

In contrary, if $C$ is very low then generated routes will be completely random even in late epochs. The effect would give an unsatisfactory probability characteristic of the output set, similar to those described in the previous sub-section. The exemplary influence of $C$ on the received results is shown in Table 2. It can be observed that despite various $\alpha$ used, the results resemble for $C$ lying in a close range.

Appropriate adjustment of $C$ is difficult with a priori knowledge only. Hence, tuning of ABAC must be empirically based.
5. Investigations

The meta-heuristic algorithms described in Section 4 were evaluated, using the experimentation system described in Section 3.2, for different set-ups, i.e., various categories of problems being solved and high-density areas (with large number of obstacles). As the reference, the sub-optimum results produced by the Bellman-Ford algorithm were taken. Meta-heuristic algorithms were executed at least 1000 times to obtain reliable output set.

5.1 Comparison of ABSA and Bellman – Ford

Results of the comparison are presented in Table 3, Table 4, and Table 5. The effects of the best and the worst combinations of parameter sets are specified. As stated in Section 4.1 no general rule for choosing the best parameters can be applied. In Table 3, the ratio $T_{stop}/T_{start}$ has lower impact (three different sets of the ratios are connected to the best result). On contrary, in Table 4, the crucial parameters are: the ratio $T_{stop} / T_{start}$, $K$ (acceptance) and $a$ (temperature drop). Moreover, it may be observed in Table 5, that the influence of $K$ is almost negligible.

Based on the obtained results of simulation experiments, it may be concluded that:

- Results produced by ABSA are significantly better for the pseudo Hamilton cycle case.
- Both algorithms (ABSA and Bellman-Ford) give comparable results for the decisive problem.
- For SPP problem, the results given by ABSA are slightly further from optimum than those given by Bellman-Ford.

The best results of Bellman-Ford and ABSA presented in Table 5 are also shown in Fig. 8. The drawback of the specialized method can be observed. In this case Bellman–Ford divides the problem into $q$ sub-problems (where $q$ is the total number of houses) and generates paths between nodes (with houses) in ascending order of the goal function increase. The ABSA outperformed Bellman-Ford algorithm because ABSA solved problem from an overall perspective.

Table 3.
The ABSA results for a decisive problem in dense area. The $pop$ is equal to 400. Bellman best for this set-up is 16300.

<table>
<thead>
<tr>
<th>$P_e=100, P_r=200, P_s=200, P_w=200, P_a=200$</th>
<th>$T_{start}$</th>
<th>$T_{stop}$</th>
<th>$a$</th>
<th>$K$</th>
<th>avg</th>
<th>std dev</th>
<th>best</th>
<th>bestBel lman</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-4}$</td>
<td>17756</td>
<td>974</td>
<td>15800</td>
<td>96.9%</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>17913</td>
<td>906</td>
<td>15900</td>
<td>97.6%</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>17821</td>
<td>889</td>
<td>15900</td>
<td>97.6%</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>200</td>
<td>.5</td>
<td>$10^{-3}$</td>
<td>19429</td>
<td>1039</td>
<td>16700</td>
<td>102.4%</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>17905</td>
<td>929</td>
<td>16700</td>
<td>102.4%</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>1200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>19786</td>
<td>896</td>
<td>18300</td>
<td>112.3%</td>
<td></td>
</tr>
</tbody>
</table>

However, as expected, during experiments, the computational time of ABSA was significantly higher (seconds for single experiment) than Bellman-Ford (milliseconds for finding solution).

Table 4.
The ABSA results for a SPP problem in dense area. The $pop$ is of 400. Bellman best (optimal solution) for this set-up is 11500.

<table>
<thead>
<tr>
<th>$T_{start}$</th>
<th>$T_{stop}$</th>
<th>$a$</th>
<th>$K$</th>
<th>avg</th>
<th>std dev</th>
<th>best</th>
<th>bestBel lman</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>12552</td>
<td>458</td>
<td>11800</td>
<td>102.6%</td>
</tr>
<tr>
<td>3000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>12489</td>
<td>377</td>
<td>11800</td>
<td>102.6%</td>
</tr>
<tr>
<td>1000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>12503</td>
<td>354</td>
<td>11800</td>
<td>102.6%</td>
</tr>
<tr>
<td>2000</td>
<td>200</td>
<td>.5</td>
<td>$10^{-3}$</td>
<td>13529</td>
<td>967</td>
<td>11900</td>
<td>103.4%</td>
</tr>
<tr>
<td>2000</td>
<td>1200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>13768</td>
<td>865</td>
<td>12000</td>
<td>104.3%</td>
</tr>
<tr>
<td>2000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-4}$</td>
<td>12733</td>
<td>524</td>
<td>12000</td>
<td>104.3%</td>
</tr>
</tbody>
</table>

Figure 8. Best paths found for the pseudo Hamilton cycle.

Table 5.
The ABSA results for a pseudo Hamilton cycles problem in dense area. The $pop$ is set to 400. Bellman best is 18700.

<table>
<thead>
<tr>
<th>$P_e=10, P_r=200, P_s=200, P_w=200, P_a=30000$</th>
<th>$T_{start}$</th>
<th>$T_{stop}$</th>
<th>$a$</th>
<th>$K$</th>
<th>avg</th>
<th>std dev</th>
<th>best</th>
<th>bestBel lman</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>19237</td>
<td>2432</td>
<td>15100</td>
<td>80.7%</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>19463</td>
<td>2566</td>
<td>15100</td>
<td>80.7%</td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>19212</td>
<td>2669</td>
<td>15200</td>
<td>81.3%</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-2}$</td>
<td>19800</td>
<td>2864</td>
<td>15500</td>
<td>82.9%</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>19735</td>
<td>2678</td>
<td>15600</td>
<td>83.4%</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>200</td>
<td>.5</td>
<td>$10^{-3}$</td>
<td>24924</td>
<td>3934</td>
<td>17900</td>
<td>95.7%</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>1200</td>
<td>.9</td>
<td>$10^{-3}$</td>
<td>45171</td>
<td>8881</td>
<td>27300</td>
<td>145.9%</td>
<td></td>
</tr>
</tbody>
</table>

5.2 Comparison of ABAC and Bellman – Ford

The ABAC was slower than the ABSA with respect to a computation time. The overall quality of the obtained results for researched cases was also less efficient.

This might be due to implementation in ABAC the simple version of the ant colony algorithm, which does not include reinforcement on movement direction choice or a pheromone dilatation.
The results of investigations are shown in Table 6, Table 7, and Table 8.

### Table 6
The ABAC (with a population of 200 elements) results for a dense area pseudo Hamilton cycle. Bellman best result is of 23640.

<table>
<thead>
<tr>
<th>gen</th>
<th>Max P</th>
<th>α</th>
<th>avg</th>
<th>std dev</th>
<th>best</th>
<th>bestBell man</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.0</td>
<td>0.1</td>
<td>35283</td>
<td>18239</td>
<td>19000</td>
<td>80.4 %</td>
</tr>
<tr>
<td>100</td>
<td>1.2</td>
<td>0.3</td>
<td>56607</td>
<td>25638</td>
<td>19850</td>
<td>83.9 %</td>
</tr>
<tr>
<td>50</td>
<td>1.0</td>
<td>0.5</td>
<td>84434</td>
<td>14806</td>
<td>54220</td>
<td>229.4 %</td>
</tr>
<tr>
<td>100</td>
<td>1.0</td>
<td>0.5</td>
<td>82742</td>
<td>13960</td>
<td>54620</td>
<td>231.3 %</td>
</tr>
<tr>
<td>80</td>
<td>1.0</td>
<td>0.5</td>
<td>81973</td>
<td>12270</td>
<td>58200</td>
<td>246.2 %</td>
</tr>
<tr>
<td>100</td>
<td>1.0</td>
<td>0.7</td>
<td>96699</td>
<td>15426</td>
<td>61570</td>
<td>260.5 %</td>
</tr>
</tbody>
</table>

### Table 7
The ABAC (with a population of 400 elements) results for a dense area decisive problem. Bellman best result is of 13800.

<table>
<thead>
<tr>
<th>gen</th>
<th>Max P</th>
<th>α</th>
<th>avg</th>
<th>std dev</th>
<th>best</th>
<th>bestBell man</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.0</td>
<td>0.7</td>
<td>18366</td>
<td>1572</td>
<td>14000</td>
<td>101.4 %</td>
</tr>
<tr>
<td>280</td>
<td>1.0</td>
<td>0.9</td>
<td>16945</td>
<td>1459</td>
<td>14100</td>
<td>102.2 %</td>
</tr>
<tr>
<td>280</td>
<td>0.5</td>
<td>0.5</td>
<td>19007</td>
<td>1852</td>
<td>14300</td>
<td>103.6 %</td>
</tr>
<tr>
<td>280</td>
<td>1.0</td>
<td>0.1</td>
<td>20845</td>
<td>2071</td>
<td>16200</td>
<td>117.4 %</td>
</tr>
<tr>
<td>280</td>
<td>1.2</td>
<td>0.3</td>
<td>19806</td>
<td>1796</td>
<td>16200</td>
<td>117.4 %</td>
</tr>
</tbody>
</table>

### Table 8
The ABAC (with a constant population of 200 elements) results for a dense area SPP. Bellman best (optimal solution) result is of 11600.

<table>
<thead>
<tr>
<th>gen</th>
<th>Max P</th>
<th>α</th>
<th>avg</th>
<th>std dev</th>
<th>best</th>
<th>bestBell man</th>
</tr>
</thead>
<tbody>
<tr>
<td>280</td>
<td>1.0</td>
<td>0.9</td>
<td>13711</td>
<td>945</td>
<td>11900</td>
<td>102.6 %</td>
</tr>
<tr>
<td>80</td>
<td>1.0</td>
<td>0.5</td>
<td>15839</td>
<td>1696</td>
<td>12100</td>
<td>104.4 %</td>
</tr>
<tr>
<td>280</td>
<td>1.0</td>
<td>0.7</td>
<td>14328</td>
<td>1165</td>
<td>12300</td>
<td>106.0 %</td>
</tr>
<tr>
<td>280</td>
<td>1.0</td>
<td>0.3</td>
<td>15706</td>
<td>1555</td>
<td>13100</td>
<td>112.9 %</td>
</tr>
<tr>
<td>280</td>
<td>1.2</td>
<td>0.3</td>
<td>15734</td>
<td>1610</td>
<td>13100</td>
<td>112.9 %</td>
</tr>
<tr>
<td>480</td>
<td>1.0</td>
<td>0.5</td>
<td>15158</td>
<td>1460</td>
<td>13300</td>
<td>114.7 %</td>
</tr>
<tr>
<td>280</td>
<td>1.0</td>
<td>0.1</td>
<td>17150</td>
<td>2000</td>
<td>13500</td>
<td>116.4 %</td>
</tr>
</tbody>
</table>

When compared with results produced by ABSA and Bellman-Ford, the ABAC reaches satisfying level only in case of the Pseudo Hamilton cycles problem. Results in this case are similar to those given by ABSA (see Table 5). The comparison of ABAC parameters used in experiments can definitely confirm that optimal value of C coefficient calculated along with (11) differs in dependence on a type of problem to be solved. It may be seen, in Table 6 and Table 7, that in most cases if higher value of C is then better results can be obtained.

However, for the Hamilton cycles problem it is not true (C<50 for close to optimum outcome).

The ABAC seems to be less efficient than the ABSA (in average 5% approx.). It should be pointed out that the standard deviation is higher for ABAC algorithm. Thus, ABAC is more resistant to the local minimum trap (especially for a higher dimensional problem) and is outputting with more information about the problem (what may be required for complex tasks).

### 6. Conclusion and Final Remarks

#### 6.1 Improvement of heuristics

Heuristic methods (algorithms) can be successfully used in solving complex tasks as presented in this paper. Despite the fact that heuristics are generally less efficient in term of computational time and quality of output for simple problems like SPP (for which deterministic Bellman-Ford method was intended to give optimum results), when being used to solve more complex tasks (pseudo Hamilton cycles or decisive problem) together with carefully adjusted parameters, they allow to shift a final result closer to optimum.

The computational time is an advantage of heuristics, when NP tasks are being encountered (frequent in modern technological problems). Due to a local minimum avoidance (paths are being generated randomly) black-box methods seem to be faster and more precise for high dimensional tasks.

Improvements applied to heuristic methods should in first place reinforce the avoiding local minima mechanism. The functionality and principles of new methods should be universal and equally efficient for different problems. The good example of a recently developed algorithm is Invasive Weed Optimization [15]. It is efficient, fast and resistant to local minima and can be applied to wide range of problems.

The next step of a black box method evolution is focus on self-adjusting algorithms. Very often, the most difficult task is the selection of proper parameters of heuristic algorithms. If such a selection could be omitted (e.g., by using self-tuning), then the method (algorithm) would be more efficient and simpler to parameter adjustment.

One of the ideas of improving efficiency is using hybrid concept. This consists in exploiting the advantages of heuristic and deterministic methods jointly. The principle is using output of a deterministic algorithm as one of elements in the first generation of a heuristic algorithm. In consequence, the final result should be equal or better than the one given by deterministic algorithm. Furthermore, it is convenient and relatively fast, if stop condition is defined as reaching specific and pre-defined goal function value which lies in the feasible solution space determined by (3), (4), (5), and (6).
6.2 Summary

In this paper, we have focused on solving path problems. For this purpose, the authors have proposed two newly created algorithms. The heuristic algorithms called ABSA and ABAC are based on heuristic approaches belonging to so-called artificial intelligence methods, in particular simulated annealing and ant colony system, respectively. Some research was made in order to compare efficiency of deterministic and heuristic algorithms for solving three categories of path problems. The analysis of results of investigations justifies such a conclusion that these algorithms seem to be promising for solving various path problems.

The main contribution of the paper consists of design and implementation of the advanced experimentation system called AreNavigator which can be used for conducting multi-aspects investigations. The core of this system is the simulator. The system has modular structure giving opportunities for creating simulation scenarios – terrains with obstacles, designing experiments in automatic manner, observing performance of three algorithms (visualization module), storing results in database, and giving possibilities for easy development, e.g., by implementation of additional algorithms solving path problems.

Recently, the system is serving as a tool to aid teaching graduate students and preparing projects in computer science and telecommunications areas in Wroclaw University of Technology.

Acknowledgement

We would like to thank Mr. Tworzydlo, student of the Department of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Technology, for preparing the first version of simulator which after development became a basic tool for making our research presented in this paper.

References