TASK PLANNING BASED ON THE INTERPRETATION OF SPATIAL STRUCTURES

Marko Švaco, Bojan Jerbić, Bojan Šekoranja

In this research, a new task planning algorithm is developed for building a desired object configuration from a given initial unordered object state. The task of the planning algorithm is to find a feasible set of actions, i.e. a finite number of discrete transformations, which can rearrange the objects into a desired ordered final state. The environment is interpreted through the position and orientation of the objects. The solution to the planning problem is proposed as a two-step method. First, a constructive heuristic generates an initial set of good solutions. The constructive heuristic uses only mutations for desirable ordered final state. The environment is interpreted through the position and orientation of the objects. The solution to the planning problem is characterized by a parallel evolutionary strategy, with the aim of spatial transformation of unordered object states into ordered object states. The algorithm can be used for solving the task planning problems represented in the two-dimensional space. Verification of the planning algorithm is done in a virtual environment.

Keywords: genetic algorithms; robotics; task planning

Algoritam planiranja zasnovan na interpretaciji prostornih struktura


Ključne riječi: genetski algoritmi; planiranje djelovanja; robotika

1 Introduction

For the new generation of industrial robots [1, 2], whose task is to work beside and in cooperation with human workers, novel intelligent control solutions and systems are intensively being developed. These control systems decrease both the needed expertise for the integration of modern robots and the programming complexity. The developed models still lack the applicability to the actual, real world, tasks and the robotics community is working intensively on finding new solutions and models for intelligent robot control. An intelligent robot control model or architecture consists of perception, planning/prediction and action modules. In this paper, a novel task planning architecture, based on the previously developed perception and classification of ARTgrid neural network [3, 4], is proposed.

Robot planning architectures and algorithms are illustrated in [5–13]. Most of the research dealing with task planning does not directly solve the path planning problems. It uses the existing path planning algorithms or certain heuristics for finding feasible and good paths from the starting to the end point. The robot task planning can be seen as a high level abstraction problem. It deals with specific problem domains, environment conditions, and specific robot kinematics.

The authors in [5] developed a robot task planning architecture in the object manipulation processes. The tasks that need to be solved by the robot are demonstrated at a symbolic level. This means that it is possible to find a larger number of geometric solutions. The developed task planning algorithm can make two-directional mappings from the symbolic robot states to the geometric workspace states. The verification of the research was done on an industrial robot with six degrees of freedom. In [6], the authors developed a robot task planning architecture for manipulation tasks. The architecture uses the first order logic to induce plans for converting initial object positions to predefined final states. In order to reduce the problem complexity, a predefined number of buffer locations are used. The system was evaluated first in a virtual environment and then on an actual, six degree-of-freedom robot. The robot was capable of manipulating three known objects assuming that the grasping points and poses were predefined.

Determining a sequence of plans based on the Adaptive Resonance Theory (ART) is part of the research done by Subaja and Tan [7]. The approach is based on the expert knowledge of the robot working domain. The authors propose a hierarchical planning architecture in a simplified blocks world. The planning architecture utilizes a multichannel ART neural network and a novel gradient classification method. The gradient classification is used to modify the standard "winner-takes-all" activation function. In the planning process, the multichannel ART NN utilizes a number of interconnected Fuzzy ART networks. The planning architecture was verified in a virtual environment.

Collaboration between a human worker and a robot in task planning and the construction of simplified 3D space structures is demonstrated in [8]. The developed robotic system has an initial assembly plan which is modified through human-robot interaction scenarios.

A robot task planning model based on touch, sound and gestures is developed in [9] and [10]. The main goal of this research is to provide an intuitive robotic application in the human-centred domains. The use case scenario consists of industrial product assembly processes...
and manipulation with known objects. The planning model is based on decomposing complex tasks into sequential task primitives.

In [11], the authors propose a novel robot task planning architecture based on ontologies. Ontologies are used for describing the specific domain of industrial manufacturing robots. Each robot contains an ontology set which comprises the robot’s capabilities and simplifies the process of manual task assignment. The robot planning module uses a sequential subtask assignment method where each subtask fulfils the previously defined preconditions.

In [12], a new genetic algorithm is developed for the task sequence assignment in an industrial robot manipulation scenario. The task set is defined as motion and a process between two known locations in the robot workspace. The novel genetic algorithm minimizes the time for completing all known subtasks by permuting the task sequences. The developed genetic algorithm controls the robot configuration changes in order to minimize the total process time. In Tab.1, a comparison between planning architectures is given.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Application</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbolic planning [5]</td>
<td>Associating symbolic and geometric working domain information</td>
<td>Manipulation planning with known objects with an industrial robot</td>
</tr>
<tr>
<td>First order logic [6]</td>
<td>Space structure construction planning</td>
<td>Construction of space structures with familiar objects with an industrial robot</td>
</tr>
<tr>
<td>ART iFALCON [7]</td>
<td>ART-based task planning architecture for software agents</td>
<td>Planning of virtual assembly tasks in a simplified blocks world</td>
</tr>
<tr>
<td>Knowledge base [9, 10]</td>
<td>Task planning and dual arm robot assembly with simplified work objects</td>
<td>A predefined assembly plan is modified on the fly through human-robot interaction scenarios</td>
</tr>
<tr>
<td>Ontologies [11]</td>
<td>Task planning based on semantic knowledge stored in ontologies</td>
<td>The initial complexity of integration of a standard industrial robot is simplified</td>
</tr>
<tr>
<td>Genetic algorithms [12]</td>
<td>Task sequence and robot kinematic configuration planning</td>
<td>Given a known task location, the set robot kinematic configurations and task sequences are optimized</td>
</tr>
</tbody>
</table>

The task planning architectures [5–11] provide only single solutions for given initial and final environment states. The genetic algorithm developed in [12] optimizes the robot task sequences but it requires a predefined set of subtasks to optimize their permutation. Genetic algorithms are commonly developed for solving combinatorial optimization problems [13].

To our knowledge, there is no unified planning architecture utilizing the evolutionary computing developed for robotic manipulation scenarios that would not only generate feasible plans but also validate and optimize them. In this research, a task planning architecture based on evolutionary computing has been developed. A constructive heuristic using a mutation only strategy inspired by specific genetic algorithms [14] was developed for generating an initial set of task plans. The task plan set is subsequently evaluated and used as an initial population for the novel genetic algorithm. The developed genetic algorithm uses the modified "swap", "remove and reinsert" and "invert" mutation operators together with a newly developed mutation operator for altering the object positions.

2 Robot task planning

The main goal of the developed task planning architecture is to generate a sequence of actions such that the sequential performance of these actions changes the robot environment and the object states from the initial unordered state \( P \) to the final ordered state \( K \), as shown in Fig. 1. To change the environment, the robot must change itself; this means that the robot changes its kinematic configuration which can have multiple solutions, as argued in [12]. This aspect is not covered in our study, so the plan assumes only the tool path.

2.1 Informal problem description

In this research, the robot behaviour is denoted as an ordered set of actions. The robot workspace is interpreted through the object type, position, and orientation in a given work plane. The task of the planning algorithm is to generate a feasible and "good enough" sequence of actions. For a given computational time period, the planning algorithm will make a solution space search and optimize the robot actions. The optimized plan will have better fitness than the initial plan but there is no optimal solution which our solution can be compared with. For that reason, we seek a "good enough" plan of actions. The actions are validated and compared through the total generated path. A use case scenario is shown in Figure 1. Seven objects are located in the scene (left) and a task plan has to be generated and carried out to achieve the final ordered object state (right).
2.2 Problem specification

The main goal of the planning algorithm is the minimization of the total travelled path in order to change the initial unordered object state $P$ into the ordered object set $K$. Three distinct subproblems can be identified with respect to the initial $P$ and the final object states $K$, as shown in Fig. 2.

In Fig. 2b), all the objects can be directly placed in their final location. For this problem, domain algorithms that solve the Travelling salesman problem (TSP) \cite{15, 16} can be used. A more complex case scenario is shown in Fig. 2c) where a certain number of objects can be directly placed on their final location by using the TSP algorithm. The remaining objects cannot be directly placed on their final location. A task plan consisting of object permutations and respective positions (initial, buffer and final positions) has to be generated. Fig. 2d) shows a use case scenario where none of the objects can be directly placed on their final locations.

A trivial solution of the robot task plan is to move all the objects such that all occupied locations are freed and then to use the greedy search or the TSP algorithm. The main downside of this approach is that the total travelled path will significantly increase.

For problem instances shown in Fig. 2c) and Fig. 2d), a novel planning algorithm based on evolutionary computing is developed.

According to \cite{17}, an initial population for a genetic algorithm can be generated using heuristics or randomly. For the task planning problem in this research, a constructive heuristic (algorithm) was developed for generating the initial population. The initial population set is generated by random mutations with a local search property. The highest mutation probability of a particular object in $P$ is in the position of the object in the final structure $K$.

### 2.3 Constructive algorithm

The constructive heuristics is initialized with the following parameters:

- 30 solutions are generated concurrently.
- After every three (3) iterations, active solutions are compared and validated; one of the 25% solutions with the lowest fitness is replaced with a clone from the 25% solutions with the highest fitness.

These 30 solutions (plans) are used as the initial population $A=\{A_1, A_2, \ldots, A_{30}\}$ of the developed genetic algorithm. The initial population of 30 solutions is experimentally verified for the given problem domain.

<table>
<thead>
<tr>
<th>ID</th>
<th>$x_0$</th>
<th>$y_0$</th>
<th>$\theta_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>96</td>
<td>161</td>
<td>-4</td>
</tr>
<tr>
<td>2</td>
<td>210</td>
<td>186</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>160</td>
<td>99</td>
<td>-80</td>
</tr>
<tr>
<td>4</td>
<td>234</td>
<td>228</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>144</td>
<td>221</td>
<td>93</td>
</tr>
<tr>
<td>6</td>
<td>184</td>
<td>144</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>137</td>
<td>153</td>
<td>-15</td>
</tr>
</tbody>
</table>
mutation operator is used to alter the position and orientation of a randomly chosen object which is not located in the final position from the active plan $A \in A$, $i=1, \ldots, 30$. The efficacy of each performed mutation is verified using the two partial fitness functions $\Phi_1$ and $\Phi_2$ combined into the global fitness function $\Phi_{1,2}$. The first partial fitness function $\Phi_1$ is given by (1).

$$\Phi_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} \text{sgn}(p_{ij} - K_{ij}) a, a = \begin{cases} 4, & \forall p_{ij} \neq K_{ij} \neq 0 \\ 1, & \forall (p_{ij} - K_{ij}) \neq 0 \\ 0, & \forall p_{ij} = K_{ij} = 0 \end{cases}$$

$\text{sgn}$ denotes the signum function and the absolute-value norm is used. $P$ denotes the matrix of the actual object states, where each identified object is discretized by a unique integer value. The process of object discretization is described in [3, 4]. $K$ denotes the matrix of the final object states where each identified object is indicated by a unique integer value. The size of matrices $P$ and $K$ is usually $200 \times 200$ pixels where $P_{ij}$ and $K_{ij}$ denote a single pixel. The sizes of individual objects in our experiments were in the $10 \div 50$ pixel range. The value of parameter $a$ is validated in the top-to-bottom order with respect to the three conditions from (1). The parameter $a$ is assigned the value from the first satisfied condition. The optimal value of $\Phi_1=0$ when all the objects in the verified plan are located in their final position. $\Phi_2$ is used for faster convergence according to the value of the "penalty" parameter $a$. The value of parameter $a$ can take only the three predefined values which define the distribution of the penalty. The highest penalty value ($a = 4$) occurs when different objects are located in the initial $P$ and the final $K$ matrix. The penalty value is zero ($a = 0$) when identical objects are located in the identical positions and orientations in the initial $P$ and the final $K$ matrix. The three empirically predefined values yield good results in the convergence of the algorithm. The second partial fitness function $\Phi_2$ calculates the Euclidian distance of all the objects in $P$ and $K$. It is given by:

$$\Phi_2 = \sum_{i=1}^{l} \sqrt{(x_i^P - x_i^K)^2 + (y_i^P - y_i^K)^2}$$

(2)

When all the objects in $P$ are located in their final position defined in $K$, $\Phi_2=0$. We do not validate the change in the orientation $\theta$ in $\Phi_2$. $\theta$ is validated through the change in discretized objects in matrices $P$ and $K$ through the partial fitness function $\Phi_1$. The total fitness function $\Phi_{1,2}$ is given as:

$$\Phi_{1,2} = g_1 \sum_{i=1}^{m} \sum_{j=1}^{n} \text{sgn}(p_{ij} - K_{ij}) a, +$$

$$+ g_2 \sum_{i=1}^{l} \sqrt{(x_i^P - x_i^K)^2 + (y_i^P - y_i^K)^2}$$

(3)

In each iteration, $\Phi_{1,2}$ is minimized and should finally converge to zero, $\Phi_{1,2} \rightarrow 0$. The weight values of partial fitness functions are given as $g_1$ and $g_2$. The weight values $g_1$ and $g_2$ are determined empirically and should be experimentally tested in different conditions. In our case, we use $g_1=1.5$ and $g_2=1$.

The mutation operator has the property of local search through the distribution of the mutation probability. The mutation probability of a particular object $O=\{1, \ldots, l\}$ has a normal distribution of positions $x$ and $y$ according to the mean $\mu_x^O = (x^K)$ and $\mu_y^O = (y^K)$, and the standard deviation $\sigma = \min \text{dim}(O)$. The $\text{min_dim}$ function calculates the minimum dimension of all the objects in the given object set.

Because of the stochastic nature of the performed mutations, the distribution of the overall fitness varies significantly. Furthermore, the number of steps required for transforming the initial space structure $P$ into the final ordered space structure $K$ ranges from 11 to 18. Fig. 5 shows the total travelled path needed to complete the planned task.

---

Fig. 3 gives an example of a generated solution having a total of 12 steps. These 12 discrete steps transform the initial unordered object set $P$ into the final space structure $K$. In each discrete step, the position and orientation of one object are altered by the constructive algorithm. The set of instructions, i.e. the task plan $A_1: P \times A_1 \rightarrow K$ given in Tab. 4, corresponds to the space structure and the object unique identifiers (ID) shown in Fig. 1. The sequence of steps define a robot task plan where an object ID is grasped from the initial location $(x_1, y_1, \theta_1)$ and placed in the final position or on a buffer location $(x_2, y_2, \theta_2)^{11}$. In Fig. 4, ten solutions are compared with respect to their fitness. All solutions start from iteration 1 when all fitness function values are validated using Eq. (3). They all have identical values because the initial position of all
objects is validated from matrix \( P \). Utilizing the local search strategy, the constructive algorithm applies repeated mutations.

<table>
<thead>
<tr>
<th>( t )</th>
<th>( ID )</th>
<th>( x_1 )</th>
<th>( y_1 )</th>
<th>( \theta_1 )</th>
<th>( x_2 )</th>
<th>( y_2 )</th>
<th>( \theta_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>184</td>
<td>144</td>
<td>23</td>
<td>131</td>
<td>85</td>
<td>44</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>160</td>
<td>99</td>
<td>-80</td>
<td>198</td>
<td>123</td>
<td>91</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>234</td>
<td>228</td>
<td>-7</td>
<td>66</td>
<td>158</td>
<td>-89</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>137</td>
<td>153</td>
<td>-15</td>
<td>106</td>
<td>246</td>
<td>-50</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>96</td>
<td>161</td>
<td>-4</td>
<td>172</td>
<td>184</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>144</td>
<td>221</td>
<td>93</td>
<td>203</td>
<td>249</td>
<td>-92</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>131</td>
<td>85</td>
<td>23</td>
<td>123</td>
<td>136</td>
<td>44</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>106</td>
<td>246</td>
<td>-15</td>
<td>125</td>
<td>199</td>
<td>-50</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>172</td>
<td>184</td>
<td>-4</td>
<td>168</td>
<td>164</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>210</td>
<td>186</td>
<td>22</td>
<td>229</td>
<td>164</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>66</td>
<td>158</td>
<td>7</td>
<td>75</td>
<td>168</td>
<td>-89</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>203</td>
<td>249</td>
<td>93</td>
<td>198</td>
<td>205</td>
<td>-92</td>
</tr>
</tbody>
</table>

**Fitness of ten solutions generated by the constructive algorithm**

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Fitness</th>
<th>Total travelled path</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.5 × 10^4</td>
<td>7000</td>
</tr>
<tr>
<td>5</td>
<td>1.5 × 10^4</td>
<td>4500</td>
</tr>
<tr>
<td>10</td>
<td>0.5 × 10^4</td>
<td>2000</td>
</tr>
<tr>
<td>15</td>
<td>0.1 × 10^4</td>
<td>1000</td>
</tr>
</tbody>
</table>

**Total travelled path of the solution set**

### 2.4 Genetic algorithm

Each solution \( \mathcal{A} \), \( i = 1, \ldots, 30 \) reached by the constructive algorithm can be presented as a permutation \( p^i \) of the task sequences. The permutation \( p^i \) defines the order in which objects in the robot workspace are manipulated (pick and place operation) from their initial position to a certain buffer position or the final position. For similar problems dealt with in [18, 19], heuristics are used to generate a feasible initial solution set. The main purpose of the developed genetic algorithm for the robot task planning is the modification of permutations of objects and their respective position and orientation in the task plan.

#### 2.4.1 Mutation operators

Regarding the plan sequence, i.e. the permutation of objects, three mutation operators for altering the object permutations are used: M1 - remove and reinset [15], M2 - swap [20], and M3 - invert [21]. The mutation operators M1-M3 are modified for the application in our problem domain. For a given permutation \( p = (1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7) \), two random positions are generated. Let the starting position be 2 and the final position 5. Then, the original mutation operators M1, M2 and M3 yield: \( p(M1)=(1 \ 3 \ 4 \ 5 \ 2 \ 6 \ 7) \), \( p(M2)=(1 \ 5 \ 3 \ 4 \ 2 \ 6 \ 7) \), \( p(M3)=(1 \ 5 \ 4 \ 3 \ 2 \ 6 \ 7) \).

The task plan, i.e. each permutation \( p^j \), can have repeated elements for which the original mutation operators M1-M3 are modified as M11, M21, M31. In the case of permutations with repeating elements \( p^j = (1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 2 ) \), a particular mutation operator M11 - M31 has to choose a partial permutation \( p^j(ij) \), with the initial element \( p^j(i) \) and the final element \( p^j(j) \), which satisfies the following conditions:

- For operator M11: \( p^j(k) = p(i) \), \( \forall k, i < k < j \)
- For operator M21: \( p^j(k) = p(i) \neq p(j) \), \( \forall k, i < k < j \)
- For operator M31: \( p^j(k) = p(i) \neq p(j) \), \( \forall k, i < k < j \)

A novel mutation operator, M4, is introduced for altering the position of objects in each task plan. A random object \( O_i \) is chosen from the current task plan and its position for the next iteration is modified according to \( (x, y) = (\Delta x, \Delta y) \). The values of \( \Delta x \) and \( \Delta y \) are generated using the mean values \( \mu_x = 0^k \) and \( \mu_y = 0^k \), and the standard deviation \( \sigma_y = \min \dim(O) \).

Mutation probability is decreased in each iteration according to parameter \( h \), which is initialized within the interval \( [0.9, 1] \): \( P(m)^{k+1} = P(m)^{k+1} \cdot h \).

#### 2.4.2 Crossover

The crossover of solutions is done by implementing the one point crossover operator [22]. Two individuals, \( p^1 \) and \( p^2 \), and the crossover point \( k \) are randomly selected from the solution set. The crossover operator alters only the permutation whereas the object positions are used from the original parent plan (solution). Two chromosomes \( p^1 = (1 \ 2 \ 3 \ 6 \ 5 \ 4 \ 2 \ 8) \) and \( p^2 = (3 \ 4 \ 2 \ 5 \ 6 \ 2 \ 8 \ 4) \) are taken as examples. In line with our problem domain, the crossover point \( k \) should be taken such that the objects after the index \( k \) in \( p^1 \) and \( p^2 \) are identical but in different permutations. Let \( k = 4 \) denote the crossover point. Then, the two individuals after the application of the crossover operator are given as:

- \( p^1 = (1 \ 2 \ 3 \ 5 \ 6 \ 2 \ 8 \ 4) \) and \( p^2 = (3 \ 4 \ 2 \ 5 \ 6 \ 4 \ 2 \ 8) \).

As task plans generated by the constructive algorithm can vary in the number of steps (Fig. 4), the crossover of plans of different lengths can also be carried. With \( p^1 \) and \( p^2 = (1 \ 2 \ 1 \ 5 \ 4 \ 3 \ 6 \ 5 \ 4 \ 8 \ 2) \) and a crossover point of \( k = 5 \),
two new individuals, \( p^{13} = (1 2 \ 3 \ 6 \ 5 \ 4 \ 8 \ 2) \) and \( p^{31} = (1 \ 2 \ 1 \ 5 \ 4 \ 3 \ 5 \ 6 \ 2 \ 8 \ 4) \), are generated.

3 Results and discussion

For the developed genetic algorithm, a convergence analysis is conducted by changing the crossover probability \( P(c) \) and the mutation probability \( P(m) \). The analysis is conducted for discrete increments of \( k = 0.1 \) in intervals \( P(c) \in [0,...,1] \), \( P(m) \in [0,...,1] \). A total number of 121 mutation and crossover probability combinations were analysed. For each pair, a total of 10 simulation runs were made. The genetic algorithm is set to stop after 300 iterations.

From the diagram in Fig. 6, with \( P(c) = 0.1 \), one can note that the highest convergence rate occurs with higher mutation probabilities, but the generated solutions do not have the overall best fitness.

A more detailed overview of the results is shown in the diagram in Fig. 7. From this 2D plot, one can identify certain areas, i.e. combinations of \( P(m) \) and \( P(c) \), that yield results with the best fitness. Furthermore, it is observed that when \( P(m) = [0...0.3] \), the total fitness is low and the lowest fitness is obtained with \( P(m) = 0 \). Multiple combinations of mutation and crossover probabilities provide good results.

Because of the specific problem domain, the mutation operator is essential for the fine tuning of object positions.

Fig. 8 shows the best solutions generated by the genetic algorithm for the use case scenario shown in Fig. 1. The plan \( A_1 \) has nine discrete steps for completing the transformation \( P \times A_1 \rightarrow K \).

For further verification, the developed genetic algorithm is compared with the greedy search algorithm. The greedy search follows the three consecutive steps to generate \( A_1 \), such that \( P \times A_1 \rightarrow K \):

- Move the closest object \( O \) to its final location if it is unoccupied. If none of the final locations is unoccupied, move to the next step.
- Choose the closest object \( O \) from \( P \) on the location where a maximum number of objects in \( K \) are located. Move the object \( O \) to the closest unoccupied location with regard to \( K \).
- If \( P \neq K \), repeat step 1; otherwise, the plan is finished.

With these two greedy rules, the closest objects that occupy other final positions in \( K \) will be moved as long as at least one final position in \( K \) becomes unoccupied. The best solution generated by the greedy search has a fitness value of \( \Phi = 2889 \).

The constructive algorithm, genetic algorithm and greedy search have been compared in 30 simulation runs with different initial conditions. The results are shown in the diagram in Fig. 9.

It can be observed that the solutions generated by the constructive algorithm are highly stochastic. This is because the constructive algorithm is used to generate the initial population for the genetic algorithm using only random mutation operators for searching the solution space. The genetic algorithm optimizes the solutions through the minimization of the total travelled path. As for the greedy search, our genetic algorithm is on average

Figure 6 Convergence analysis of the crossover probability \( p_c = 0,1 \) and different values of the mutation probability \( p_m \).

Figure 7 For each combination of \( P(m) \) and \( P(c) \), 10 simulation runs are calculated. In the plot, a mean value is shown for a total of 1210 simulations.
18% better, i.e. gives solutions with 18% better fitness. Because of the stochastic nature of the constructive algorithm, the solution generated in the 12th iteration is better than the solution generated by the greedy search.

**Figure 8** Best solution generated by the genetic algorithm \( \Phi = 2497 \)

**Figure 9** Comparison between the solutions generated by the constructive algorithm, genetic algorithm and greedy search

### 4 Future work

The previously developed ARTgrid neural network [3, 4] for the classification of space structures and the planning algorithm developed in this research will be applied to the novel robot control model. ARTgrid will be used for classifying the ordered object states and also for associating an unordered object set with the previously classified space structures. The planning algorithm developed in this paper will be used as the second part of the robot control model for the planning and performing tasks. The experimental part of our research is planned to be carried out on an industrial robotic arm with an integrated vision system. The algorithm for generating robot trajectories [23] could be used to optimize the movements of the robot joints.

### Acknowledgements

The authors would like to acknowledge the support of the European Regional Development Fund through the project RONNA. The authors would also like to acknowledge the support of the Croatian Scientific Foundation through the research project ACRON - A new concept of Applied Cognitive ROboties in clinical Neuroscience.

### 5 References


[20] Pongcharoen, P.; Stewardson, D. J.; Hicks, C.; Braiden, P. M. Applying designed experiments to optimize the performance of genetic algorithms used for scheduling complex products in the capital goods industry. // Journal of Applied Statistics. 28, 3-4(2001), pp. 441-455. DOI: 10.1080/02664760120034162

