ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20160118150332

TASK PLANNING BASED ON THE INTERPRETATION OF SPATIAL STRUCTURES

Marko Švaco, Bojan Jerbić, Bojan Šekoranja

Original scientific paper

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 making an initial population of state transitions. A genetic algorithm is developed for optimizing the initial set of solutions. The genetic algorithm 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

Izvorni znanstveni članak

U ovom istraživanju razvijen je novi algoritam planiranja za transformaciju početnog neuređenog stanja objekata u uređeno konačno stanje. Zadatak algoritma planiranja je pronaći mogući niz djelovanja kojima se početno stanje okoline, kroz konačan broj diskretnih transformacija, može dovesti u zadano konačno stanje. Stanje okoline tumači se kroz položaj i orijentaciju objekata. Zadatak planiranja rješava se u dva koraka. Razvijena je konstruktivna heuristika pomoću koje se dobiva početni skup rješenja. Konstruktivna heuristika koristi mutacije za generiranje početne populacije. Genetski algoritam je razvijen za optimizaciju početnog skupa rješenja. Genetski algoritam karakteriziran je usporednom evolucijskom strategijom za pronalaženje rješenja, s ciljem prostorne pretvorbe neuređenog stanja objekata u uređeno, ograničen na dvodimenzionalnu interpretaciju radnog prostora. Verifikacija algoritma planiranja napravljena je u virtualnom okruženju.

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 1 Comparison between planning architectures

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

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).

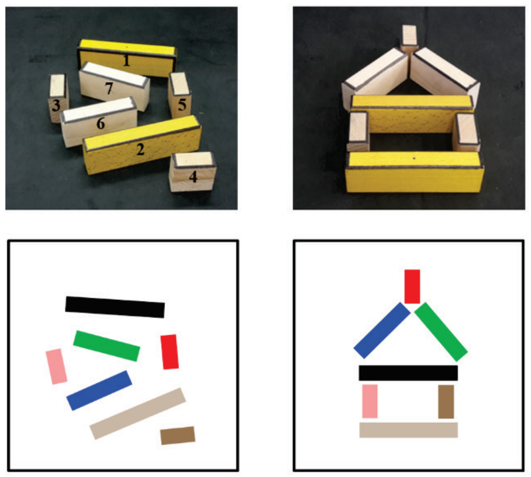

Figure 1 An initial object set (left) and the final object set (right)

Table 2 Object matrix ***P*** of the initial object state

ID$_0$	x_0	y_0	θ_0
1	96	161	−4
2	210	186	22
3	160	99	−80
4	234	228	7
5	144	221	93
6	184	144	23
7	137	153	−15

Table 3 Object matrix ***K*** of the final object state

ID$_T$	x_T	y_T	θ_T
1	168	164	0
2	229	164	0
3	198	123	91
5	198	205	−92
4	75	168	−89
7	125	199	−50
6	123	136	44

The values x_0, y_0 and θ_0 denote the initial position and orientation of an object while the values x_T, y_T and θ_T denote the final position and orientation. Tab. 2 shows the object matrix $\boldsymbol{P}$ containing the information on all the objects in their initial positions while Tab. 3 shows the object matrix $\boldsymbol{K}$ of the final object state.

For solving the task planning problem, two subproblems are identified. We identify the action sequencing problem and the object position assignment problem. The action sequencing problem is identified as a permutation (with repeating elements) which defines the sequence in which the objects are manipulated. The problem related to the object positions (and orientations) is defined for every task step.

The planning problem: Generate a discrete time and space set of activities $\mathcal{A}$ over n known objects which are in an unordered initial structure $\boldsymbol{P}$. The activity set $\mathcal{A}$ has to transform $\boldsymbol{P}$ into an ordered spatial structure $\boldsymbol{K}$. The generated activity set must not result in object collisions in any instance of $A_i \in \mathcal{A}$. For solving this planning problem, we assume the following:

- Each solution $A_i \in \mathcal{A}$ gives $\boldsymbol{P} \times A_i \rightarrow \boldsymbol{K}$.
- All objects are located in a reference plane.
- Time and space can be discretised.

Definition 1: An activity is defined as a "pick and place" operation that consists of picking an object, linear approach and retract motions, and moving and placing the object in its final position in $\boldsymbol{K}$ or a buffer location. In a finite time interval $t = [t_i, t_{i+j}]$ ($i \geq 0, j > 0, j > i$), a robot can do only one activity.

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 $\boldsymbol{P}$ into the ordered object set $\boldsymbol{K}$. Three distinct subproblems can be identified with respect to the initial $\boldsymbol{P}$ and the final object states $\boldsymbol{K}$, 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) [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.

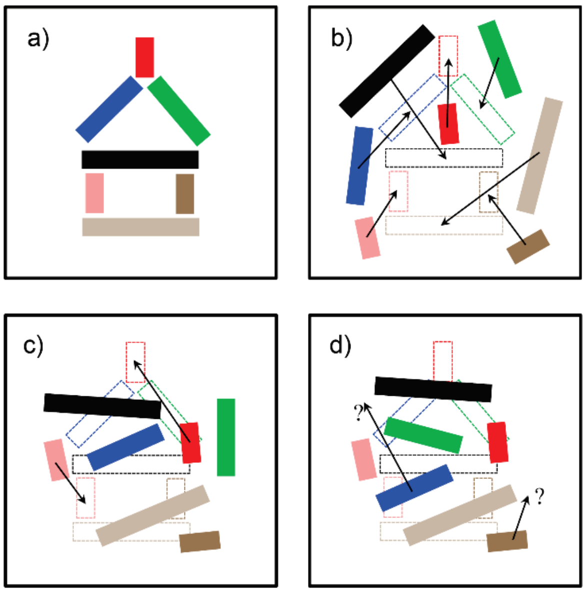

Figure 2 a) Final space structure ***K***, b) trivial case scenario c) certain objects can be directly placed in their final positions d) none of the objects can be directly placed in their final positions

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 [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 $\boldsymbol{P}$ is in the position of the object in the final structure $\boldsymbol{K}$.

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 $\mathcal{A} = \{A_1, A_2, \ldots, A_{30}\}$ of the developed genetic algorithm. The initial population of 30 solutions is experimentally verified for the given problem domain. A 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 $\boldsymbol{A}_i \in \mathcal{A}$, i=1,…, 30. The efficacy of each performed mutation is verified using the two partial fitness functions Φ_1 and Φ_2 combined into the global fitness function $\Phi_{1,2}$. The first partial fitness function Φ_1 is given by (1).

$$\Phi_1 = \sum_{i=1}^{m}\sum_{j=1}^{n} \operatorname{sgn}\left(\left|P_{ij} - K_{ij}\right|\right)a,\ a = \begin{cases} 4, \forall P_{ij} \neq K_{ij} \neq 0 \\ 1, \forall\left(P_{ij} - K_{ij}\right) \neq 0 \\ 0, \forall\left(P_{ij} - K_{ij}\right) = 0 \end{cases} \quad (1)$$

Sgn denotes the signum function and the absolute-value norm is used. $\boldsymbol{P}$ 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]. $\boldsymbol{K}$ denotes the matrix of the final object states where each identified object is indicated by a unique integer value. The size of matrices $\boldsymbol{P}$ and $\boldsymbol{K}$ is usually 200×200 pixels where P_{ij} and K_{ij} denote a single pixel. The sizes of individual objects in our experiments were in the 10 ÷ 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 Φ_1=0 when all the objects in the verified plan are located in their final position. Φ_1 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 $\boldsymbol{P}$ and the final $\boldsymbol{K}$ matrix. The penalty value is zero (a = 0) when identical objects are located in the identical positions and orientations in the initial $\boldsymbol{P}$ and the final $\boldsymbol{K}$ matrix. The three empirically predefined values yield good results in the convergence of the algorithm. The second partial fitness function Φ_2 calculates the Euclidian distance of all the objects in $\boldsymbol{P}$ and $\boldsymbol{K}$. It is given by:

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

When all the objects in $\boldsymbol{P}$ are located in their final position defined in $\boldsymbol{K}$,Φ_2=0. We do not validate the change in the orientation θ in Φ_2. θ is validated through the change in discretized objects in matrices $\boldsymbol{P}$ and $\boldsymbol{K}$ through the partial fitness function Φ_1.The total fitness function $\Phi_{1,2}$ is given as:

$$\begin{aligned}\Phi_{1,2} &= g_1\sum_{i=1}^{m}\sum_{j=1}^{n} \operatorname{sgn}\left(\left|P_{ij} - K_{ij}\right|\right)a, + \\ &+ g_2\sum_{i=1}^{l}\sqrt{\left(x_i^P - x_i^K\right)^2 + \left(y_i^P - y_i^K\right)^2}\end{aligned} \quad (3)$$

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, …, i} has a normal distribution of positions x and y according to the mean $\mu_i^x = (x^K)$ and $\mu_i^y = (y^K)$, and the standard deviation $\sigma = \min\_\dim(O)$. The 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 $\boldsymbol{P}$ into the final ordered space structure $\boldsymbol{K}$ ranges from 11 to 18. Fig. 5 shows the total travelled path needed to complete the planned task.

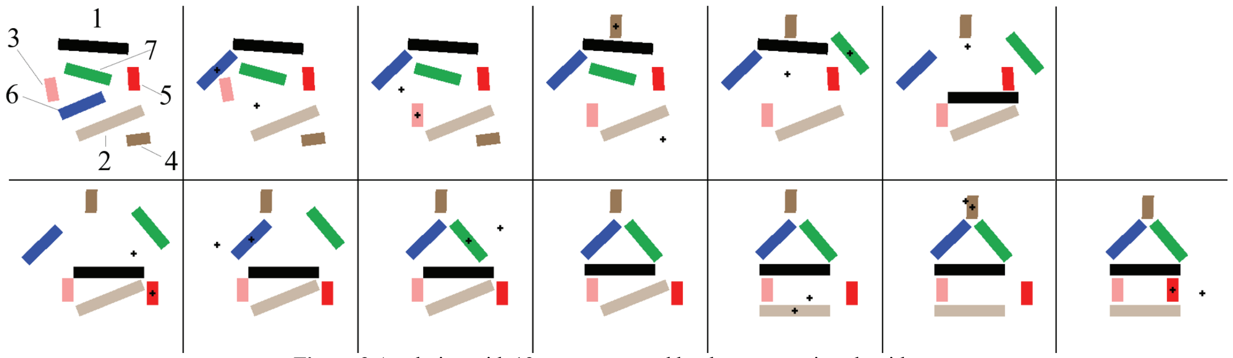

Figure 3 A solution with 12 steps generated by the constructive algorithm

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 $\boldsymbol{P}$ into the final space structure $\boldsymbol{K}$. 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 $\mathbf{A}_1$: $\boldsymbol{P}\times\boldsymbol{A}_1\rightarrow\boldsymbol{K}$ 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 $(x_1, y_1, \theta_1)^t$ and placed in the final position or on a buffer location $(x_2, y_2, \theta_2)^{t+1}$.

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 4 A set of instructions generated by the constructive algorithm corresponding to Fig. 1

t	*ID*	x_1	y_1	θ_1	x_2	y_2	θ_2
1	6	184	144	23	131	85	44
2	3	160	99	−80	198	123	91
3	4	234	228	7	66	158	−89
4	7	137	153	−15	106	246	−50
5	1	96	161	−4	172	184	0
6	5	144	221	93	203	249	−92
7	6	131	85	23	123	136	44
8	7	106	246	−15	125	199	−50
9	1	172	184	−4	168	164	0
10	2	210	186	22	229	164	0
11	4	66	158	7	75	168	−89
12	5	203	249	93	198	205	−92

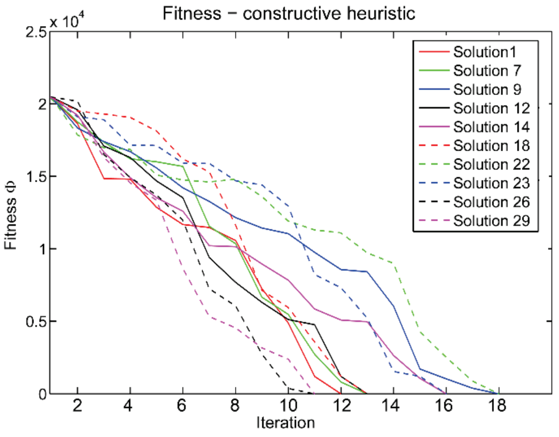

Figure 4 Fitness of ten solutions generated by the constructive algorithm

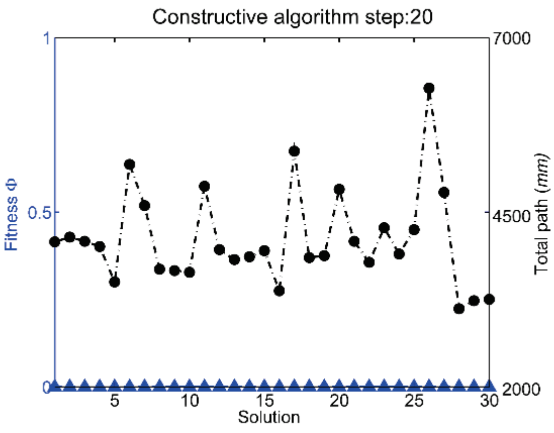

Figure 5 Total travelled path of the solution set

2.4 Genetic algorithm

Each solution $\mathcal{A}_i \in \mathcal{A}$, i=1,…,30 reached by the constructive algorithm can be presented as a permutation $\mathbf{p}^i$ of the task sequences. The permutation $\mathbf{p}^i$ 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 reinsert [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 $\mathbf{p}$ = (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: $\mathbf{p}$(M1)=(1 3 4 5 2 6 7), $\mathbf{p}$(M2)=(1 5 3 4 2 6 7), $\mathbf{p}$(M3)=(1 5 4 3 2 6 7).

The task plan, i.e. each permutation $\mathbf{p}^i$, can have repeated elements for which the original mutation operators M1-M3 are modified as M1^1, M2^{1,} M3^1. In the case of permutations with repeating elements $\mathbf{p}^I$ = (1 2 3 4 5 6 7 2) , a particular mutation operator M1^1 - M3^{1}has to choose a partial permutation $\mathbf{p}^I(i\text{-}j)$, with the initial element $\mathbf{p}^I(i)$ and the final element $\mathbf{p}^I(j)$, which satisfies the following conditions:

- For operator M1^1 $p(k) \neq p(i), \forall k, i < k < j$
- For operator M2^1 $p(k) \neq p(i) \neq p(j), \forall k, i < k < j$
- For operator M3^1 $p(k) \neq p(i) \neq p(j), \forall k, i < k < j$

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 $(x,y)^{t+1}$=(Δx, Δy). The values of Δx and Δy are generated using the mean values μ_x=x^K and μ_y=y^K, and the standard deviation σ_{xy}=min_dim(O).

Mutation probability is decreased in each iteration according to parameter h, which is initialized within the interval $h \in [0{,}9,...,1]$: $P(m)^{t+1}$=$P(m)^{t+1} \cdot h$.

2.4.2 Crossover

The crossover of solutions is done by implementing the one point crossover operator [22]. Two individuals, $\mathbf{p}^1$ and $\mathbf{p}^2$, 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 $\mathbf{p}^1$=(1 2 3 6 5 4 2 8) and $\mathbf{p}^2$=(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 $\mathbf{p}^1$ and $\mathbf{p}^2$ 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:

$$^{c}\mathbf{p}^1 = (1\,2\,3\,5\,6\,2\,8\,4) \text{ and } ^{c}\mathbf{p}^2 = (3\,4\,2\,6\,5\,4\,2\,8)\,.$$

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 $\mathbf{p}^1$ and $\mathbf{p}^3$ = (1 2 1 5 4 3 6 5 4 8 2) and a crossover point of k = 5, two new individuals, ${}^{c}\mathbf{p}^{13} = (1\,2\,3\,6\,5\,4\,8\,2)$ and ${}^{c}\mathbf{p}^{31} = (1\,2\,1\,5\,4\,3\,5\,6\,2\,8\,4)$, 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 $P(\mathrm{m})$. 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 $P(\mathrm{m})$ 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 $\boldsymbol{A}_1$ has nine discrete steps for completing the transformation $\boldsymbol{P}\times\boldsymbol{A}_1\rightarrow\boldsymbol{K}$.

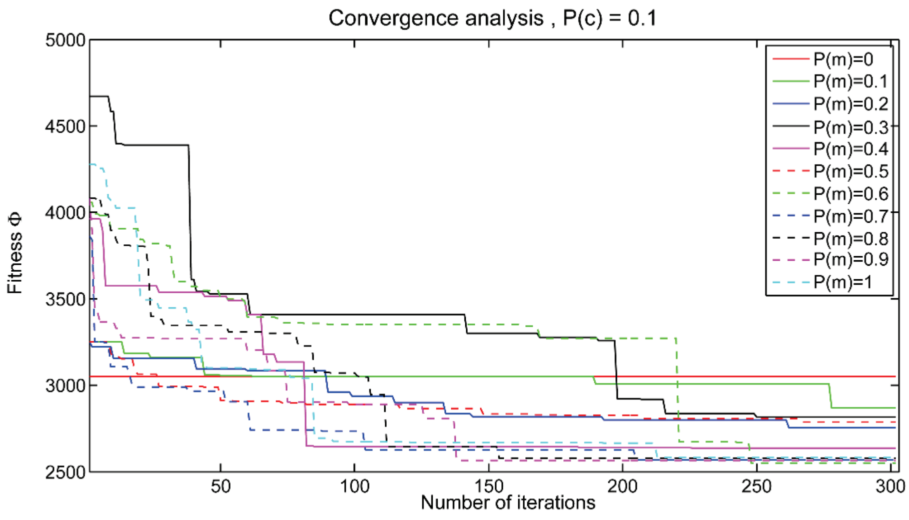

Figure 6 Convergence analysis of the crossover probability p_c = 0,1 and different values of the mutation probability p_m

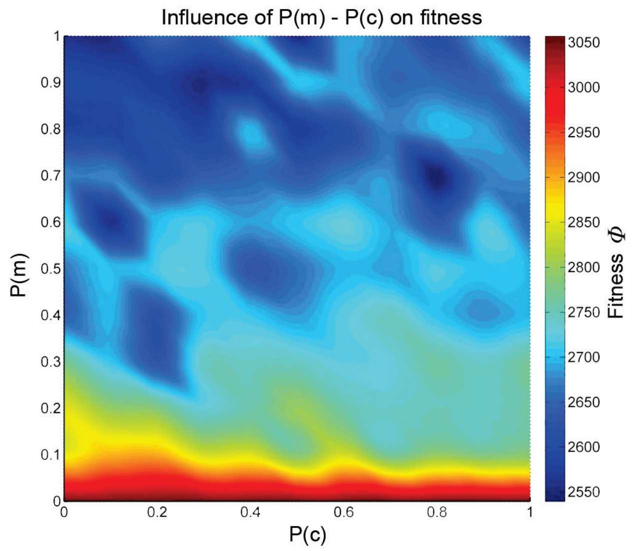

Figure 7 For each combination of $P(m)$ and $P(c)$, 10 simulation runs are calculated. In the plot, a mean value is shown for a total of 1210 simulations.

For further verification, the developed genetic algorithm is compared with the greedy search algorithm. The greedy search follows the three consecutive steps to generate $\boldsymbol{A}_i$, such that $\boldsymbol{P}\times\boldsymbol{A}_i\rightarrow\boldsymbol{K}$:

- 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 $\boldsymbol{P}$ on the location where a maximum number of objects in $\boldsymbol{K}$ are located. Move the object O to the closest unoccupied location with regard to $\boldsymbol{K}$.
- If $\boldsymbol{P}\neq\boldsymbol{K}$, repeat step 1; otherwise, the plan is finished.

With these two greedy rules, the closest objects that occupy other final positions in $\boldsymbol{K}$ will be moved as long as at least one final position in $\boldsymbol{K}$ 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

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.

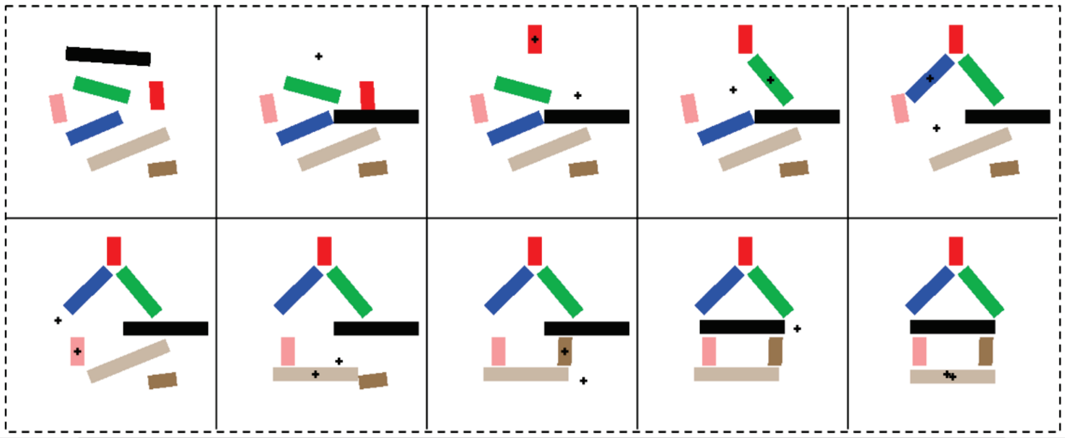

Figure 8 Best solution generated by the genetic algorithm Φ=2497

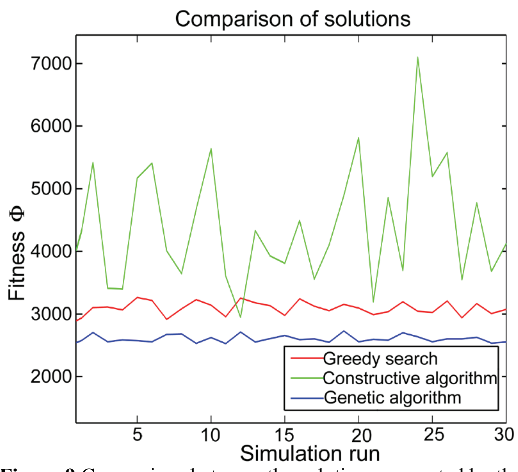

Figure 9 Comparison between the solutions generated by the constructive algorithm, genetic algorithm and greedy search

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 RObotics in clinical Neuroscience.

5 References

[1] Guizzo E.; Ackerman, E. The rise of the robot worker. // Spectrum, IEEE. 49, 10(2012), pp. 34-41.

[2] Jerbić, B.; Nikolić, G.; Chudy, D.; Švaco, M.; Šekoranja, B. Robotic application in neurosurgery using intelligent visual and haptic interaction. //International Journal of Simulation Modelling. 14, 1(2015), pp. 71-84. DOI: 10.2507/IJSIMM14(1)7.290

[3] Švaco, M.; Jerbić, B.; Šuligoj, F. Autonomous Robot Learning Model Based on Visual Interpretation of Spatial Structures. // Transactions of FAMENA. 38, 4(2014), pp. 13-28.

[4] Švaco, M.; Jerbić, B.; Šuligoj, F.ARTgrid: A Two-Level Learning Architecture Based on Adaptive Resonance Theory. // Advances in Artificial Neural Systems, 2014, pp. 1-9. DOI: 10.1155/2014/185492

[5] Dearden, R.; Burbridge, C. Manipulation planning using learned symbolic state abstractions. //Robotics and Autonomous Systems. 62, 3(2014) pp. 355-365. DOI: 10.1016/j.robot.2013.09.015

[6] Ekvall, S.; Kragic, D. Robot learning from demonstration: a task-level planning approach. // International Journal of Advanced Robotic Systems. 5, 3(2008), pp. 223-234. DOI: 10.5772/5611

[7] Subagdja, B.; Tan, A.-H. iFALCON: A neural architecture for hierarchical planning. // Neurocomputing. 86(2012) pp. 124-139. DOI: 10.1016/j.neucom.2012.01.008

[8] Zhang, J.; Knoll, A. A two-arm situated artificial communicator for human-robot cooperative assembly. // IEEE Transactions on Industrial Electronics. 50, 4(2003), pp. 651-658. DOI: 10.1109/TIE.2003.814767

[9] Lenz, C.Context-aware human-robot collaboration as a basis for future cognitive factories. // Technische Universität München, 2011.

[10] Bannat, A.; Bautze, T., Beetz, M.; Blume, J.; Diepold, K.; Ertelt, C.; Geiger, F.; Gmeiner, T.; Gyger, T.; Knoll, A.; Lau, C.; Lenz, C.; Ostgathe, M.; Reinhart, G.; Roesel, W.; Ruehr, T.; Schuboe, A.; Shea, K.; Stork., I.; Wersborg, G.; Stork, S.; Tekouo, W.; Wallhoff, F.; Wiesbeck, M.; Zaeh, M. F.Artificial Cognition in Production Systems. // IEEE Transactions on Automation Science and Engineering. 8, 1(2011), pp. 148-174. DOI: 10.1109/TASE.2010.2053534

[11] Stenmark, M.; Malec, J. Knowledge-based instruction of manipulation tasks for industrial robotics. // Robotics and Computer-Integrated Manufacturing. 44, (2016). DOI: 10.1016/j.rcim.2014.07.004

[12] Zacharia, P. T.; Aspragathos, N. A. Optimal robot task scheduling based on genetic algorithms. //Robotics and Computer-Integrated Manufacturing. 21, 1(2005), pp. 67-79. DOI: 10.1016/j.rcim.2004.04.003

[13] Puljić K. An Evolutionary Algorithm Based on Repeated Mutations for Solving the Capacitated Vehicle Routing Problem. //Journal of Computing and Information Technology. 20, 1(2012). DOI: 10.2498/cit.1002019

[14] Shiu, K. L.; Szeto, K. Y. Self-adaptive Mutation Only Genetic Algorithm: An Application on the Optimization of Airport Capacity Utilization. // in Intelligent Data Engineering and Automated Learning – IDEAL 2008, vol. 5326, C. Fyfe, D. Kim, S.-Y. Lee, and H. Yin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 428-435.

[15] Larranaga, P.; Kuijpers, C. M. H.; Inza, I.; Dizdarevic, S. Genetic algorithms for the travelling salesman problem: a review of representations and operators. // Artificial Intelligence Review. 13 (1999), pp. 129-170. DOI: 10.1023/A:1006529012972

[16] Rego, C.; Gamboa, D.; Glover, F.; Osterman, C. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. // European Journal of Operational Research. 71, 3(2011), pp. 427-441. DOI: 10.1016/j.ejor.2010.09.010

[17] Puljić, K.; Manger, R. A distributed evolutionary algorithm with a superlinear speedup for solving the vehicle routing problem. // Computing and informatics. 31, 3(2012), pp. 675-692.

[18] Gultekin, G. K.; Ogucu, M. O.; Sayginer, E.; Saranli, A. A genetic algorithm based solution to constrained component placement of a mobile robot. // International Symposium onInnovations in Intelligent Systems and Applications (INISTA), 2012, pp. 1-6. DOI: 10.1109/INISTA.2012.6246933

[19] Glover, F.; Kochenberger, G. A. Handbook of metaheuristics. Boston: Kluwer Academic Publishers, 2003. DOI: 10.1007/b101874

[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

[21] Hwang, H. S. An improved model for vehicle routing problem with time constraint based on genetic algorithm. // Computers & Industrial Engineering. 42, 2-4(2002), pp. 361-369. DOI: 10.1016/S0360-8352(02)00033-5

[22] Mitchell, M. An introduction to genetic algorithms. Cambridge, Mass.: MIT Press, 1996.

[23] Sidobre, D.; He, W.Online task space trajectory generation. // Workshop on Robot Motion Planning: Online, Reactive, and in Real-time 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, 2012, pp. 121-128.

Authors' addresses

Marko Švaco, PhD
University of Zagreb,
Faculty of Mechanical Engineering and Naval Architecture,
Ivana Lučića 5, 10000 Zagreb, Croatia
E-mail: marko.svaco@fsb.hr

Bojan Jerbić, Full professor
University of Zagreb,
Faculty of Mechanical Engineering and Naval Architecture,
Ivana Lučića 5, 10000 Zagreb, Croatia
E-mail: bojan.jerbic@fsb.hr

Bojan Šekoranja, PhD
University of Zagreb,
Faculty of Mechanical Engineering and Naval Architecture,
Ivana Lučića 5, 10000 Zagreb, Croatia
E-mail: bojan.sekoranja@fsb.hr