INTECH
open science | open minds

OPEN ACCESS
ARTICLE

International Journal of Advanced Robotic Systems

Co-Evolutionary Algorithm for Motion Planning of Two Industrial Robots with Overlapping Workspaces

Regular Paper

Petar Curkovic[1,*], Bojan Jerbic[1] and Tomislav Stipancic[1]

1 Faculty of Mechanical Engineering and Naval Architecture, Zagreb, Croatia
* Corresponding author E-mail: petar.curkovic@fsb.hr

Received 29 Jun 2012; Accepted 9 Nov 2012

DOI: 10.5772/54991

© 2013 Curkovic et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract A high level of autonomy is a prerequisite for achieving robotic presence in a broad spectrum of work environments. If there is more than one robot in a given environment and the workspaces of robots are shared, then the robots present a dynamic obstacle to each other, which is a potentially dangerous situation. This paper deals with the problem of motion planning for two six-degrees-of-freedom (DOF) industrial robots whose workspaces overlap. The planning is based on a novel hall of fame - Pareto-based co-evolutionary algorithm. The modification of the algorithm is directed towards speeding-up co-evolution, to achieve real-time implementation in an industrial robotic system composed of two FANUC LrMate 200*i*C robots. The results of the simulation and implementation show the great potential of the method in terms of convergence, robustness and time.

Keywords Evolutionary Algorithms, Motion Planning, Multi Robot Systems, Optimization

1. Introduction

Symmetrically distributed limbs are a trait of higher primates. This fact can be observed as an optimum achieved by a long evolutionary process, or as a result of an optimization process. The limbs enable humans, for example, to complete tasks, which require the use of both hands and cannot be completed by using only one hand. In this line of thought, placing two robots next to each other enables the robots to be more flexible, to help each other and mutually increase their scope of work. Moreover, by placing robots so that their workspaces overlap, their DOFs are shared. Thus, the flexibility of the system increases. A standard industrial robot has a six DOF structure, whereas a human hand has 27 degrees of freedom in total, with 8 of them located in the wrist. The disparity is obvious and the consequence is that today robots, even the most sophisticated ones, have problems with opening a door, for example.

One could ask why a robot or a robotic hand with more than six degrees of freedom has not been built. It has, but the problems associated with controlling such a system are complex and limit its application. Problems with accuracy and repeatability limit implementation even more seriously [1, 2]. Building robots with a higher number of degrees of freedom is not the only way to increase their work scope. We adopt an anthropomatic

approach, where a system consists of more than one robot, but the kinematics of the robots is relatively simple. This way, the control of each robot is simpler, but favourable traits of high flexibility, increased scope of work and autonomy result from the combination of several simple structures.

This paper deals with the problem of motion coordination for two standard six DOF industrial robots whose workspaces are shared (Figure 1).

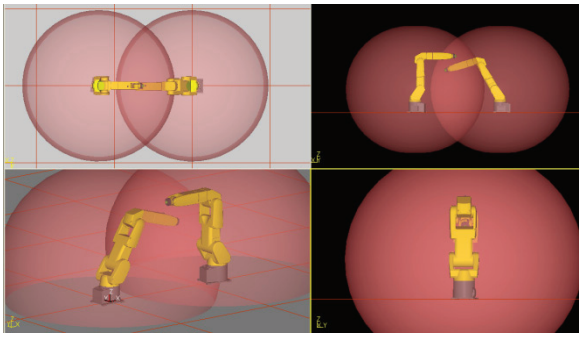

Figure 1. Robot setup. Spheres indicate the corresponding workspaces.

The idea behind arranging robots this way is to increase the work scope of the robotic system with respect to the set of actions that can be performed by two separate robots. It is immediately possible to conclude that two robots in such a setup cannot work completely independently. The occurrence of collision is almost certain if one robot does not take into account the motion of the other one. The simplest way to tackle this problem is to ensure a safety envelope for each robot and to ensure that the other robot never enters the safety envelope of the first one. This is actually a very common way to solve this problem in industrial applications but limits the performance of the robots as a system working together and achieving synergy. Another common approach is to give a role of "master" to one of the robots and declare the other robot a "slave". The slave robot constantly tracks the position of the master and adapts its movement so that no interference between the robots' links occurs. The problem is that such a scenario results in intermittent motion of the robots and time delays. There are also approaches that utilize simple breakaway screws or external shock sensors. In such applications, collision actually occurs, with all the physical consequences, replacement and restart procedures and the results in a production delay.

The issues mentioned above have resulted in an increased focus by the research community on finding suitable solutions for autonomous motion planning for a single robot, or in recent times, in the multi-robot domain. A comprehensive overview on robot motion planning can be found in [3]. If a path planning problem is presented in the optimization context, robust optimization techniques, such as evolutionary algorithms [4], swarm intelligence concepts [5] etc. have proven suitable, even though the problem is NP-complete and PSPACE-hard even in its simplest form. Its complexity increases exponentially with the dimensions of the robot's configuration space [6]. In [7] a special genetic algorithm (GA) for optimized robot trajectories is proposed. The main characteristics of this algorithm are the use of dynamic chromosomes structures and a modified crossover operator called an analogous crossover. The goal of the proposed GA is to minimize the accumulative deviation between the actual and the desired path. A multi-objective GA for the evolution of joint-space strings for manipulator configurations is proposed in [8]. Five indices, namely the travelling distance of the manipulator joints, the joint velocity, the Cartesian distance, the Cartesian velocity and energy are used to qualify the evolving trajectories. In [9] the preliminary results of the Constructive Solid Geometry-based approach to path planning of multiple robot arms are presented. The authors use a two phase GA to obtain a plan for robotic arms by using a strategy that combines the exploration of free space, while looking for the target position from each previously explored area.

The majority of papers dealing with the application of evolutionary algorithm-based methods for path planning consider significant simplifications, such as a single robot in an environment without obstacles or a robot surrounded by a set of stationary obstacles.

An approach to the path planning of a dual arm reconfigurable robot is presented in [10]. The paper deals with a SCARA configuration dual arm robot controlled by a single controller. The algorithm is based on the configuration space exploration for the given initial and final configuration of two hands, carried out in a horizontal plane.

A coupled dual arm system is a system composed of two robots with a point of contact between them. The contact is usually realized between the end-effectors of the two robots. In this scenario, impedance control-based methods may be used to solve the coordination of the two robots [11].

This paper presents a co-evolutionary approach to solving the path planning of two robots that share their workspace and are controlled by separate controllers. The input variables are the current positions of the two robots, given by the end-effector coordinates and the desired positions of the two robots, again given by the reference point coordinates in a reference coordinate system. The main characteristics of the proposed algorithm are: real value encoded chromosomes with joint angles on the loci, two coevolving populations (each population represents a set of potential solutions to path

planning for one robot), the Pareto-based selection used for the evaluation of a multi objective inter-population fitness function and the hall of fame procedure for the preservation of the cross-population best collaborators (since the quality of each individual from one population depends on individuals from the other population).

The performance of the algorithm is tested through a number of simulation experiments and compared to the generic co-evolutionary algorithm without the Pareto-based selection and the hall of fame preservation. After the fine-tuning of the parameter space, the algorithm is implemented in a real robotic system, at this point limited to planning in a vertical or horizontal plane.

2. The proposed co-evolutionary algorithm

The purpose of this paper is not to outperform the existing heuristic approaches or to find better solutions to the problem of dual arm robot motion planning. Its purpose is to show how co-evolutionary algorithms may be efficiently applied to this problem and to describe how to do it.

Co-evolutionary algorithms offer great potential for concurrent multiagent domains [12 - 14]. Concepts of co-evolution were also successfully coupled with other heuristic optimization techniques [15 - 17] to solve complex optimization problems when search-spaces are connected. The main problems reported concerning co-evolutionary algorithms are their game-theoretic background and resulting pathologies, namely cyclic dynamics, loss of fitness gradient and evolutionary forgetting [18].

1. for population $p_s \in P$*, all populations*
1.1 ***Initialize*** *population* p_s
2. for population $p_s \in P$*, all populations*
2.2 ***Evaluate*** *population* p_s
3. t:=0
4. do
4.1 for population $p_s \in P$*, all populations*
4.1.1 ***Select parents*** *from population* p_s
4.1.2 ***Generate offspring*** *from parents*
4.1.3 ***Select collaborators*** *from P*
4.1.4 ***Evaluate*** *offspring with collaborators*
4.1.5 ***Select survivors*** *for new population* P_s
4.2 t:=t+1
until ***Terminating criteria*** *is met.*

Figure 2. Pseudo-code for a sequential co-evolutionary algorithm

The motivation for using co-evolutionary algorithms to solve the defined problem is the distributed nature of the problem itself. Prior to this research, we experimented with standard evolutionary algorithms and realized [19] that the explicit fitness function for the evaluation of a system composed of a single robot with several optimization criteria becomes increasingly complex. The consequence is that the time required for finding feasible solutions is unacceptably long if the algorithm is to be implemented with real robots.

Co-evolution is by nature a distributed process, which can be applied to two separate controllers, making the computation parallel and thus reducing the time required for finding feasible solutions. Moreover, the definition of co-evolution is the adaptation of one species triggered by a change in the other species, which is directly applicable to the adaptation of the behaviour of one robot to the behaviour of another one. The abstract pseudo-code of a co-evolutionary algorithm is presented in Figure 2. The interaction between the two coevolving populations naturally mimics the solution of the problem described in this paper.

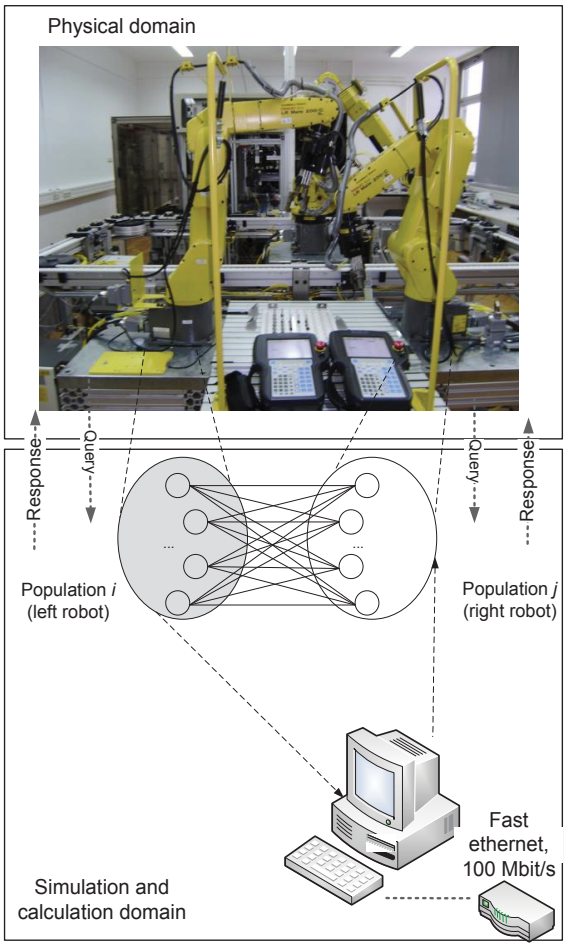

Figure 3. System architecture

The setup used for the purpose of experimental evaluation in this paper is given in Figure 3. The two robots are arranged so as to ensure the overlapping of corresponding workspaces.

Motion planning is based on two coevolving populations. Each population consists of a set of potential solutions to

the problem of path planning from the given initial to the given final configuration for the left and right robot, respectively.

An individual in the population encodes a set of transitional configurations of the robot. At this time the planning is executed in the vertical or horizontal plane, limiting the search space only to two DOF for each robot. An individual in a population is represented as a chromosome containing a real-valued vector in the joint space of the robot (Equation 1):

$$\left[\left\{q_{11}^{(\Delta t,G)},..,q_{ij}^{(\Delta t,G)}\right\},\left\{q_{11}^{(2\Delta t,G)},..,q_{ij}^{(2\Delta t,G)}\right\},..,\left\{q_{11}^{((n-2)\Delta t,G)},..,q_{ij}^{((n-2)\Delta t,G)}\right\}\right] \quad (1)$$

where i denotes the robot i= 1, 2, j is the number of DOF (since the problem is reduced to the *planar* plane, j = 2 but generally it is expandable to the full six DOF space), Δt is the sampling time between two consecutive configurations, q is the angle between the link and the positive x-axis and G is the current generation.

Adopting the form of an individual in the population according to (1), which is a discrete form of an individual resulting in discrete information given to the robot controller, there is a trade off between the granularity of the discretization and the real time computational requirements. There is an important question – how can we decide what is the appropriate length of the chromosome representing the individual and defining the granularity of discretization?

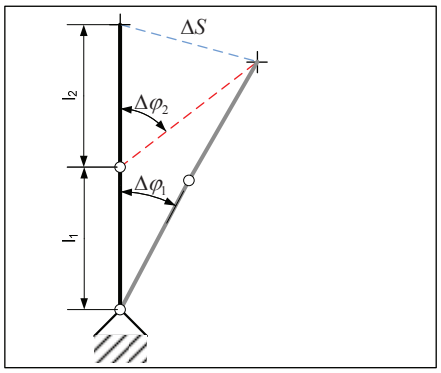

Figure 4. Calculation of the chromosome length

It is assumed for this calculation that the robot moves as shown in Figure 4: if the second joint of the robot is static, the joint displacement of the first joint is $\Delta\varphi_1$; if the first joint of the robot is static, the joint displacement of the second joint is $\Delta\varphi_2$. In both cases the distance travelled by the reference point equals ΔS.

The following formula can be derived: $\Delta\varphi_1 \cdot (l_1 + l_2) = \Delta\varphi_2 \cdot l_2 = \Delta S$, where l_1 and l_2 are the lengths of the robot links. If T is assumed to be sampling time, the distance travelled by the reference point is $\Delta S = V_{\max} \cdot T$ and the angles travelled can be obtained from Equation 2:

$$\Delta\varphi_{1,\max} = \frac{V_{\max} \cdot T}{l_1 + l_2};\ \Delta\varphi_{2,\max} = \frac{V_{\max} \cdot T}{l_2}, \quad (2)$$

where V_{max} is for the maximum velocity of the Tool Centre Point (TCP). For this calculation we will assume that V_{max} = 5 m/s, T = 0.01 s and l_1 = l_2 = 0.25 m. The angles calculated are: $\Delta\varphi_1 \le 11.5^\circ, \Delta\varphi_2 \le 23^\circ$. Thus, the discretization space is bounded. The calculation of the chromosome length L is as follows:

$$L \ge 2 \cdot \frac{\Delta\overline{\varphi}_i^G}{\Delta\varphi_{1,\max}},\ l \in \mathbb{N} \ \wedge\ l \ne 0 \quad (3)$$

where $\Delta\overline{\varphi}_i^G = \left|\varphi_i^{(n-2,G)} - \varphi_i^{(1,G)}\right|$ is the difference between the initial and the final angle of rotation of the i-th robot link and L is the length of the chromosome rounded up to the first larger integer. Number 2 in equation (3) is for the pairs of values in the chromosome, two angles for one configuration. The worst case, when the initial and the final configuration of the robot are on the boundaries of the upper semi-plane, will actually never happen in reality. The resulting chromosome length is L = 32, which means 16 configurations are needed at the upper boundary.

Since the evaluation of populations composed of chromosomes whose length is 32 is computationally complex and therefore is impossible to perform in real time, we keep the length at L<10, which is experimentally proven to be achievable in real time, calculating a lower number of intermediate points. A simple interpolation procedure is developed to reconstruct the missing configurations, based on the polynomial of at least the third order. In that case, solutions are obtainable in real time.

3. Fitness evaluation

It is obvious that fitness evaluation has to take into account the number of collisions between the links of the robots and the trajectory length. Both of these criteria have to be minimized. After the initial simulation it was determined that these two parameters are not enough and that additional criteria are required to obtain satisfactory behaviour of the robots, so velocity profile and total joint rotation angle are added. By adding these two parameters smooth motions of the robots, free of unnecessary joint rotations were obtained. The criteria are described in details as follows:

Collision penalty

The collision penalty, depending on a collision between Robot 1 and Robot 2 in corresponding configurations, is given by:

$$C_1 = \sum_{k=1}^{k=n-2} C_k, C_1 \to \min \tag{4}$$

where:

$$C_k = \begin{cases} 1 \text{ if } R_1 \text{ and } R_2 \text{ collide in i}^{\text{th}} \text{ generation} \\ 0 \text{ otherwise.} \end{cases} \tag{5}$$

Since the initial and final configurations are known in advance and by definition, a collision cannot exist, to save calculation time, these two configurations are omitted from the search, thus k = n-2. This criterion is the most important one. In order to accept the solution and to implement it in real robots, the value has to be equal to zero, which physically means that there is no collision between the two potential solutions. It is important to note that for the evaluation of the expression defined in Eq. (4), individuals of both co-evolving populations are required. This is where the co-evolution takes place, since the quality of an individual from population 1 depends on all the individuals from the population 2. An individual from the first population has to be evaluated against every individual from the other population. The total number of collisions involving an individual from the first population is summed and then this number must be minimized. This sum is not necessarily required to equal zero, rather it is important to find one good match from the other population for the given individual from the first population, for which the collision number equals zero. This is a cross-population criterion and is thus the most complex. Also, it cannot be computed independently in each population or executed in parallel.

The total distance of the end-effector movement is as follows:

$$C_2 = \sum_{k=1}^{k=n} dist\left(p_j, p_{j-1}\right), C_2 \to \min \tag{6}$$

where p_j is the intermediate position of the end-effector and *dist* is the Euclidean distance between two consecutive intermediate positions. The optimal value of this criterion is known in advance and equals the length of the straight line that connects the initial and final end-effector positions. This is obtainable only in a limited set of scenarios where no obstacle is present in the environment of the robot. For all other cases, $C_2 > C_{2opt}$. This is an inter-population criterion and possibly can be calculated in parallel, similarly to the following two criteria.

Total angle of rotation

Since the robots are redundant systems even in this simple form of 2 *DOF*, resulting in the possibility of reaching the same point in the space in elbow-up and elbow-down configurations, criteria that minimize the total distance are not enough. Additionally, it is necessary to minimize the total rotation angle, to ensure no oscillations between the elbow-up and elbow-down configurations occur. The following expression defines the total angle for one joint of one robot. Each robot's total angle should be minimized:

$$C_3 = \sum_{i=1}^{n} \left| \alpha_i - \alpha_{i-1} \right| \to \min, C_3 \to \min \tag{7}$$

where α_i is the angle between the link of the robot and the positive horizontal axis in the discrete time step is i.

End-effector velocity distribution

To enable an even distribution when going through points p_j of the end-effector, the distance between two adjoining points in unit time should be equal.

$$C_4 = \left\{ dist\left(p_j, p_{j-1}\right)_{\max} - dist\left(p_j, p_{j-1}\right)_{\min} \right\} \to \min, C_4 \to \min \tag{8}$$

where *dist* is calculated as a Euclidean distance between the previous and current position of the end-effector. The optimal value for this criterion is known in advance to be 0, which means that all the points that are gone through are equally distant from each other. The fitness itself is calculated as a sum of criteria C_1 – C_4 and has to be minimized.

Regarding other evolutionary operators, a simple single point crossover showed good results. Standard values of crossover in the range $p_c \in [0.6, 0.7]$ were used. When considering mutation, the situation is not so simple. An higher rate of mutation was shown to be beneficial in the early phase of the evolution, whereas at the later stage it was beneficial to decrease mutation probability. Details regarding this effect and its impact on convergence were analysed in methods MI-MVIII. The mutation was performed using the following expression:

$$q_{ij}^{(\Delta t, G+1)} = q_{ij}^{(\Delta t, G)} + rand \in \left(0, \pi / 5\right] \Big| q_{ij}^{(\Delta t, G+1)} < \pi \tag{9}$$

The mutation operator replaces one allele with a given probability using Eq. 9.

The size of the populations is also an important part of successful convergence. It is said that one part of the problem of any evolutionary algorithm is creating the algorithm itself [20]. The other part of the problem is fine-tuning the algorithms parameters, whose interconnections are nonlinear, so often a significant effort is necessary to make the algorithm work well. This surely holds also for the co-evolutionary algorithms that have even more complex natures compared to standard evolutionary algorithms [21]. The problem with large populations is the time required to find satisfactory

solutions. Note that in this problem, the best possible solution is not always searched for, since time is a critical factor. Rather any feasible solution that solves the problem in terms of collision free trajectory is accepted if there is not enough time to continue searching. Therefore, the sizes of the populations are 100 individuals. To increase the speed of the convergence, we modify the co-evolution to approach the real-time usability of the system, developing methods of selecting the best collaborators based on a combination of the Pareto front exploration and the hall of fame creation to save the best collaborators from the co-evolving populations.

The Pareto front exploration aims to identify the best individuals from a given population. The individuals forming the Pareto front are also known as non-dominated solutions. This front is a 4 -dimensional one, partial fronts are illustrated in Figure 5. It is a very significant and positive trait of the proposed algorithm that it is able to identify the Pareto fronts.

We want to employ this trait to keep the best solutions that form the Pareto front and allow these solutions unconditioned transition to the following generation. These solutions might serve as a good seed to direct the evolution in the desired direction in a shorter period of time.

This is analogous to the elitism in a simple evolutionary algorithm, but different in the way that the elitism is not directly employable in co-evolutionary algorithms since the performance of each individual from one population depends on each individual from the other, co-evolving population, as already described.

It is obvious that the proposed algorithm can find solutions on the Pareto front in an early phase of evolution, after only 30 generations (Figure 5). This is an advantage of the algorithm, with the consequence of an increased likelihood of finding the global-optimal area of the multi objective search space. The red dots indicate individuals on the Pareto front, or non-dominated solutions, whereas blue dots indicate inferior, dominated solutions. Red, non-dominated solutions are the highest quality ones, i.e., the ones that are kept in the hall of fame until replaced with fitter ones.

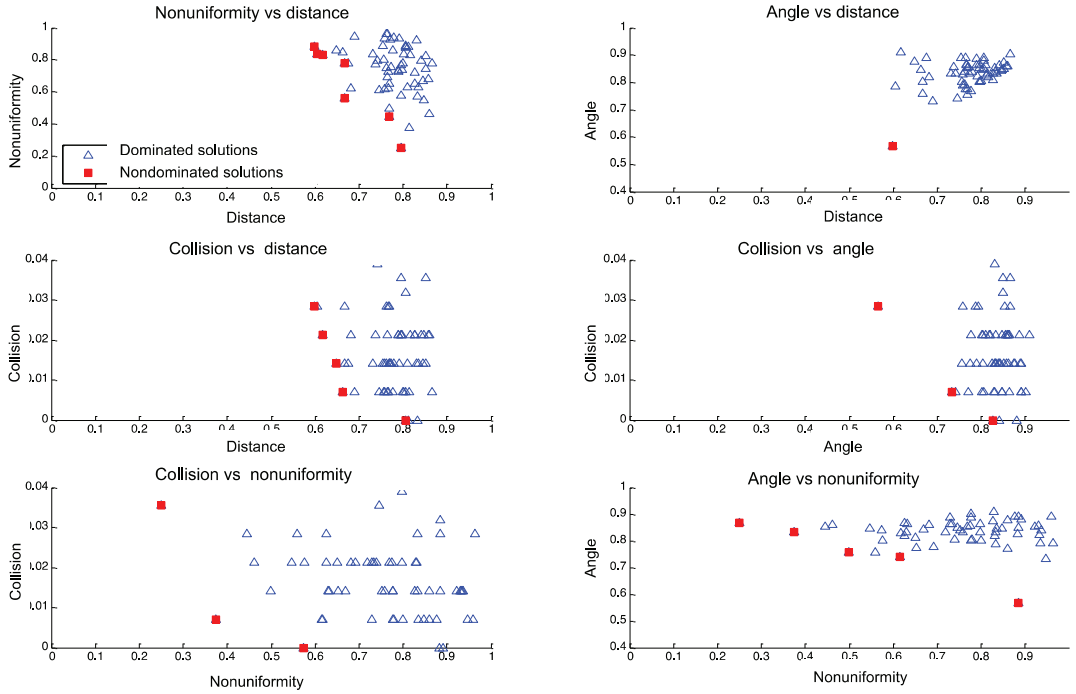

Figure 5. Pareto fronts in an early phase of evolution (G=30) for various pairs of evaluation criteria

The hall of fame is an operator that determines how many of the individuals are to be kept and directly copied to the following generation. The depth of the hall of fame is a percentage of the individuals in respect to the size of the original population that are directly copied to the next generation. The members of the hall of fame are only replaced when better (fitter) solutions on the Pareto front are identified.

The problem analysed in this paper represents a multi objective optimization problem with conflicting criteria. All the criteria are to be minimized, but they are conflicting in nature.

The time required to find a feasible solution becomes a critical parameter once the algorithm is implemented in the real physical system. In this paper, the convergence

time is accepted if it is in the range of <10s. However, since the algorithm is heuristic, it is possible that no feasible solution is found in the predefined time range (<10s). The good thing is that it is easy to check whether the algorithm has converged, based on the critical criterion of a collision. If no convergence has occurred, and collision is present, the algorithm generates new populations and the search starts again. Since planning is done only once, at the beginning of the motion, a pause of 10 *s* max can occur during the calculation. Often, the time is significantly shorter than the maximally acceptable time, i.e. 10 s. Actually, every motion solution is acceptable if the resulting number of collisions equals to zero. This solution might not be optimal in terms of all optimization criteria, but if time is critical, it can be adopted and executed.

Regarding the number of generations required for convergence, we have compared several variations of the standard approach based on the roulette wheel selection, methods I-IV, Fig. 6. To increase the speed of convergence, we introduced the Pareto-based selection. The idea is to keep not only the best collaborators, as pairs of values from both populations in the hall of fame, but also to include the whole Pareto front of the current generation for evaluation. The methods based on the Pareto domination are the labelled methods V-VIII in Fig. 6. It is visible that a significant impact on the speed of convergence is achieved by introducing the new selection method.

The differences between methods are as follows: I – IV are individuals of the standard roulette wheel selection process, with different values of evolutionary parameters, such as size of populations, mutation and crossover probabilities.

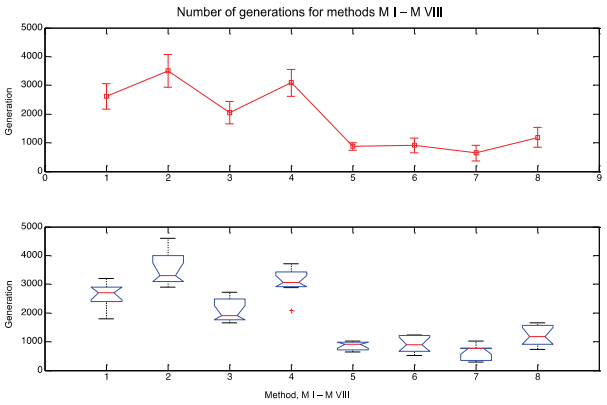

Figure 6. Impact of the modified co-evolution on the convergence speed

Methods V–VIII are based on the Pareto selection, with different parameters of population sizes, crossover and mutation operator values. The methods are evaluated with 20 runs for each method and the corresponding results are illustrated, with standard deviation, in the upper part of Fig. 6 and the box-plot in the lower part of the same figure. We experimented with the depth of the hall of fame procedure. Namely, if the hall of fame is not changed during the process of evolution, an opposite effect in terms of convergence speed can occur, since the number of good individuals tends to grow. A consequence is that more calculations per generation are required. In an early phase of evolution, this is not important since the hall of fame is empty, or contains a small number of solutions. The parameter depth of the hall of fame indicates the memory effect, which determines after how many generations the new members will start to replace the old ones. For our purposes, we obtained the best results when a hard limit of 15% of the population size was set to limit the size of the hall of fame. It is important to note that the size of the front changes during evolution, since the global optimum for some criteria might be found, which means the front is actually reduced to a single point.

Regarding the mutation, as mentioned already, we experimented with a simple, constant mutation probability. It was determined that problems with a constant mutation rate occur in the later phase of evolution, when higher mutation rates have a devastating impact on the individuals. To overcome this issue the following procedure was designed and implemented. Eq. 10: higher mutation rates are beneficiary in an early phase of a search in order to quickly sample the search space. Once "good" parts of the search space are identified, the mutation should decrease in order to not destroy good individuals.

$$p_m = 1 - 0.9 \cdot \frac{t}{G} \tag{10}$$

where p_m is mutation probability, t is the current generation or iteration step and G is them maximally acceptable number of generations. To sum the evolutionary parameters for the methods yielding the best results (methods MV-MVIII) the Pareto front is copied to the next generation. The size of the front is limited to 15% of the size of the original population.

The mutation rate is variable, according to Eq. 10, when roulette wheel selection is implemented and individuals are encoded as real-valued vectors with joint angles at the loci. Since the number of evaluations per generation is large, it is important to identify the operation that is most-time consuming. It was identified that the collision check procedure is the most time-consuming operation. At first, we used a simple method of calculating intersections between robot links defined as lines. For a population of size 100 and two robots with two links each, the number of calculations for only these criteria is 100x100x2x2 and the same number of calculations has to

be carried out for each discrete time step. This is a large number with a high impact on calculation time. We have analysed the problem and were able to significantly reduce this time by introducing dimensionless parameters and by transposing the calculation to only the angle of the rotation space. The procedure is as follows: Dimensionless parameters μ and ν are introduced. Their values define the existence of the intersection point *P*. The intersection exists if and only if $\{\mu,\nu\} \in (0,1]$.

The dimensionless parameters are defined as $\overrightarrow{O_1P} = \mu\vec{b}; \overrightarrow{O_2P} = \nu\vec{d}$, as illustrated in Fig. 7.

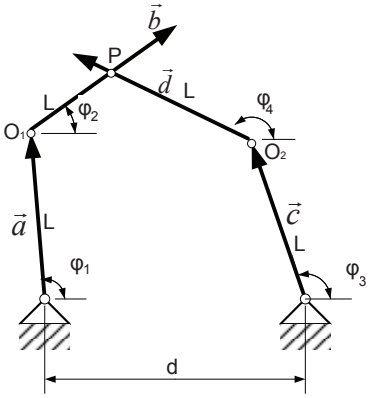

Figure 7. Analysis of intersection *P* between the top links of the robots

Solving the following equation:

$$\vec{a} + \mu\vec{b} = \nu\vec{d} + \vec{c} + d\vec{i} \tag{11}$$

we can obtain the values for parameters μ and ν :

$$\mu = \frac{\sin(\varphi_3 - \varphi_4) - \sin(\varphi_1 - \varphi_4) - \xi \cdot \sin(\varphi_4)}{\sin(\varphi_2 - \varphi_4)}, \tag{12}$$

$$\nu = \frac{\sin(\varphi_2 - \varphi_1) + \sin(\varphi_3 - \varphi_2) - \xi \cdot \sin(\varphi_2)}{\sin(\varphi_2 - \varphi_4)}.$$

Parameter $\xi = d / L$

The consequence of this procedure is a significantly less complex calculation than the calculation of the intersections of a set of lines, with the beneficial consequence of increasing the speed of the calculation. Analogously, the collisions between two arbitrary links of the robots can be checked.

4. Implementation and experimental results

The proposed algorithm was examined through multiple simulations in Matlab (Fig. 8 and 9) and finally on the real robots. Fig. 8 presents the result of the motion of the two robots in the vertical plane with 2 DOF each. The red cross stands for the end position of the end-effector of the corresponding robot, whereas the black circle stands for the start position of the end-effector. The configurations of the robots are chosen so as to lead to collision if each robot follows the shortest path from the start position of the end-effector to the end position of the end-effector. In this setup, each robot has to adapt its motion in order to avoid the collision with the other robot occupying the workspace.

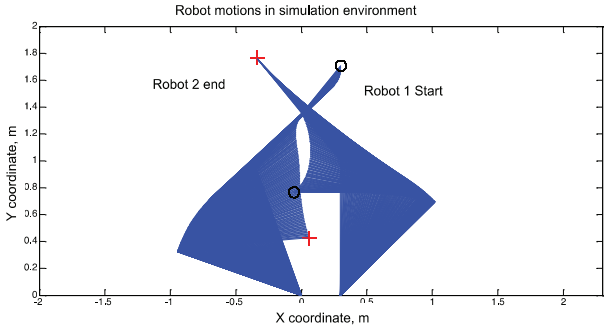

Figure 8. Motion of two robots in the Matlab simulation environment

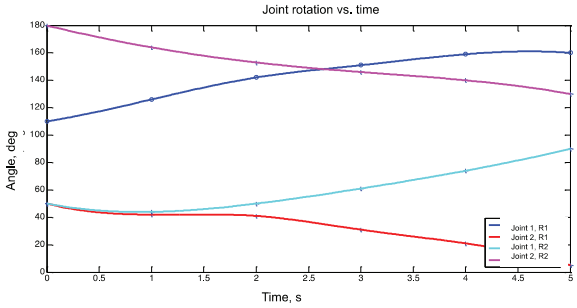

Figure 9. Interpolated joint rotation values for discrete values obtained by the co-evolutionary algorithm

This is the reason why a blank space is visible between the set of points describing consecutive positions of the end-effector positions of each robot. The picture itself describes the trail of the robot, starting from the initial configuration and ending with the final one.

Fig. 9 describes the interpolated values for the discrete joint rotation values obtained as a result of the co-evolutionary algorithm. The interpolation is performed using a polynomial of the third order to ensure second order derivability and thus smooth transitions. Also, it is necessary to check the gradient of the interpolating polynomial to physically enable the robot to perform the motion.

The PC that runs Matlab performs all the calculations upon receiving the actual and desired configurations of the robot. To enable communication protocols in the system, we assigned the following IP addresses: Robot 1 (left robot): 192.168.123.26; Robot 2: 192.168.123.25 and PC: 192.168.123.40. Two communication channels were opened. The first channel is for sending data from a robot to the PC. In that case, the robot has the role of the server,

while the PC is a client receiving the desired data. The second channel is for sending the response from the PC to the robot controller.

Now the PC has the role of the server and the robot controller is the client receiving the desired data. Messages that are sent are coded and can have different forms. They indicate whether the motion is possible or not (i.e., whether the intersection of the robot links in the initial or final configurations). If the motion is possible, a list of coordinates and going-through positions of the end-effector is sent.

It is important to ensure that the time delay between the start of the motion of both robots is kept to minimum. This is done through directly connected I/O signals between the two robot controllers.

The role of these signals is to trigger the motion at a desired moment in both robots. The algorithm was tested for various scenarios: different initial and final conditions, one robot moving while the other one is waiting, stationary obstacles present in the workspace, etc.

Fig. 10 and 11 present the final checks prior to implementation in the real robot setup. The pictures are from the Roboguide physical simulator developed by Fanuc Robotics. Fig. 10 is for two Lr Mate robots, for a simple scenario when one robot is stationary. In this case, the problem boils down to avoiding a stationary obstacle.

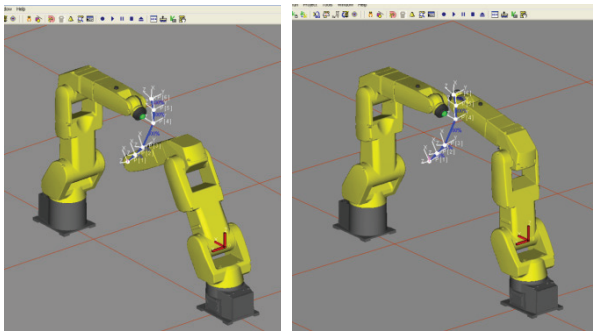

Figure 10. Motion execution in the Roboguide physical simulator

This is a simple scenario, but a problem was noticed after implementation in the physical simulator. Namely, since the robots are defined as a set of lines, the collision can actually occur due to the physical dimensions of the real robot, even if the solution from the simulator is collision-free. To tackle this problem, we described each link of the robot as a set of two boundary lines.

A circle with a variable radius size parameter is defined around the TCP to enable tuning according to the current tool the robot is carrying (as illustrated in Fig. 11). Since the robots are to have various tools attached to the flange, we immediately noticed the problem of the fixed geometry of the robot as implemented in Matlab.

The three trajectories in Fig. 11, implemented in the M10 robot, in this case illustrate how the safety factor works, forcing the robot to move away from a potentially dangerous zone by various amounts depending on the robot's current setup. In order to implement the algorithm in our M10 robot, we had to adapt the lengths of the links and global coordinate systems in the Matlab simulation environment. Note also that this motion was executed in the horizontal plane, providing robots with only the desired position of the end-effector and using robots internal kinematics model to calculate everything else.

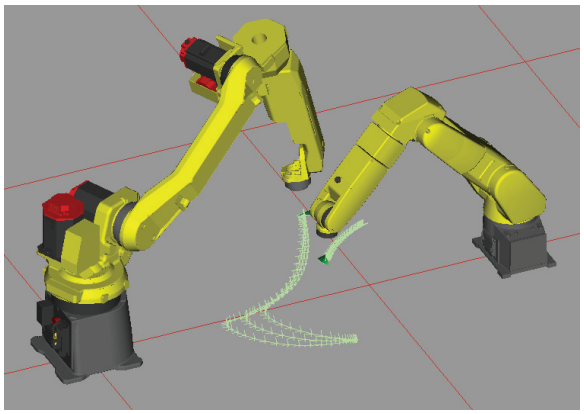

Figure 11. Safety factor for the end-effector

The distance between the lines describing the robot and the sides of the rectangle were experimentally determined to fit the configurations of the robots used in our laboratory. This way the problems of inconsistencies between the model and the real robot were eliminated.

5. Conclusions

In industrial environments where several autonomous robots work in a shared environment, it is essential to find feasible, collision-free motions.

In this paper, we have presented a method based on co-evolutionary algorithms that finds optimal or near–optimal trajectories for two industrial robots that share a workspace. Exact values for the granularity of calculation, through chromosome length, are calculated, based on the physical limitations of the robotic system used for experimental verification. This approach is scalable in terms of adapting it to different dimensions of robots. Also, it is general and implies the possibility of adding new degrees of freedom or new robots to the system. Standard approaches to co-evolutionary algorithms are implemented and evaluated and modifications, including the Pareto-based selection and the transition of collaborations to the new generations, are presented. The methods are compared and experimentally verified. The result of the modifications is the increase in speed of the convergence in the case of the modified algorithm. A formal model analysis of the dual arm robot is performed

to simplify the time-consuming operation of collision detection. The computational complexity is reduced by means of a collision check conducted only by corresponding angles of rotation, instead of using the exhaustive line intersections procedure.

The algorithm developed in this research is expected to serve as a low–level procedure ensuring the safe motion of robots requested to perform high–level tasks. To solve the problem, we have developed a simulation environment that analyses movements of simplified robot models. The next step is implementation in the physical simulation environment, where the interaction between physical constraints and the simplified model occurs and the tuning of the model can be performed to satisfy real-world constraints. Finally, the full implementation is performed on two Fanuc Lr Mate 200*i*C robots.

We anticipate further work in adding more robots and more degrees of freedom to the model. The model is developed to be scalable and allows the inclusion of additional features, although important questions regarding the computational power are expected. The main bottleneck of the approach remains its computational complexity, despite the effort to include algorithmic simplification.

A possible way to overcome this problem is to apply the co-evolutionary process to the distributed computer architecture and to make independent evolutionary operators perform in parallel.

At this point in the research, planning is performed only once, at the beginning of the motion of the two robots. This is fine if nothing unexpected occurs during the motion of the two robots. If one robot unexpectedly stops during the motion, there would be a possibility of a collision with the other robot, since the robots are not aware of each other after the motion starts. This is a limitation, but future work will consider permanent motion sampling for the two robots. If a possibility of collision is detected, the planning routine can be invoked more than once to avoid collisions after initial planning.
Features for web-based, real time monitoring and control of such a complex robotic setup, see for example [22, 23] are also considered for future research. This would further increase the safety of a system tending to be of a high level of autonomy.

6. Acknowledgments

The authors would like to thank the Ministry of Science, Education and Sports of the Republic of Croatia for financial support of the project *Autonomous multiagent automated assembly* through grant no. 120-1201948-1941. The authors would also like to thank the Editor and anonymous reviewers for their valuable comments and suggestions, which were helpful in improving the paper.

7. References

[1] Skorc, G., Cas, J., Brezovnik, S. & Safaric, R. (2011) Position Control with Parameter Adaptation for a Nano-Robotic Cell, Strojniški vestnik - Journal of Mechanical Engineering, Vol. 57, No. 4, 313-322.

[2] Meneses, J., Castejon, C., Corral, E., Rubio, H. & Garcia-Prada, J. C. (2011) Kinematics and Dynamics of the Quasi-Passive Biped "PASIBOT", Strojniški vestnik - Journal of Mechanical Engineering, Vol. 57, No. 12, 879-887.

[3] Latombe, J. C. (1991). Robot motion planning, Kluwer Academic Publishers, Boston.

[4] Abo-Hammour, Z. S.,, Alsmadi, O.,., Batineh, S.I.,., Al-Omari, M.A., & Affach, N.. (2011) Continuous Genetic Algorithms for Collision-Free Cartesian Path Planning of Robot Manipulators, Int. Journ. of Adv. Rob. Sys., Vol. 6.

[5] Ćurković, P. & Jerbić, B. (2007) Honey-bees optimization algorithm applied to a path planning problem, Int. Jour. of Sim. Modeling. VI, 154-165.

[6] Nearchou, A. (1998) Path planning robot using genetic heuristics, Robotica, Vol.16, 575-588.

[7] Davidor, Y. (1991) Genetic Algorithms and Robotics; A Heuristic Strategy for Optimization, WorldScientific, Singapore.

[8] Solteiro Pires, E. J., Tenreiro Machado, J. A. & Moura Oliveira, P. B. (2004) Robot Trajectory Planning Using Multi-objective Genetic Algorithm Optimization, Proc- of the GECCO, LNCS.

[9] Venegas Montes, H. A. & Raymundo Marcial-Romero, J. (2009) An Evolutionary Path Planner for Multiple Robot Arms, Evo Workshops, LNCS, Springer, Heidelberg.

[10] Fey, Y., Fuqiuang, D. & Xifang, Z., (2004) Collision-free motion planning of dual-arm reconfigurable robots, Robotics and Computer-Integrated Manufacturing, Vol. 20, 351.

[11] Surdilovic, D., Yakut, Y., Nguyen, T. M., Pham, X. B., Vick, A. & Martin, M. (2010) Compliance Control with Dual-Arm Humanoid Robots: Design, Planning and Programming, IEEE-RAS Int. Conf. on Humanoid Robots, Nashville, TN, USA.

[12] Panait, L. & Luke, S. (2007) Biasing Coevolutionary Searches for Optimal Multiagent Behaviors, IEEE Transactions on Evolutionary Computation, Vol. 10, No. 6

[13] Ćurković, P. & Jerbić, B. (2010) Dual-Arm Robot Motion Planning based on Cooperative Coevolution, Emerging trends in technological innovation, Heidelberg, Springer Verlag, 169-179

[14] Curkovic, P., Jerbic, B. & Stipancic, T. (2013) Coordination of robots with overlapping workspaces based on motion co-evolution, International Journal of Simulation Modeling. In press. To appear in Vol. 11, No. 1.

[15] Sinha, A. K., Zhang, W. J. & Tiwari, M. K. (2012) Co-evolutionary immuno-particle swarm optimization with penetrated hyper-mutation for distributed inventory replenishment, Engineering Applications of Artificial Intelligence, Available online 16th February 2012.
[16] Kou, X., Liu, S., Zhang, J. & Zheng, W. (2009) Co-evolutionary particle swarm optimization to solve constrained optimization problems. Computers & Mathematics with Applications, Vol. 57, No. 11–12, 1776-1784
[17] Zhao, Q., F., Hammami, O., Kuroda, K. & Saito, K. (2000) Cooperative co-evolutionary algorithms - how to evaluate a module? IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks, 150-157
[18] Kou, X., Liu, S., Zhang, J. & Zheng, W. (2009) Co-evolutionary particle swarm optimization to solve constrained optimization problems, Computers & Mathematics with Applications, Vol. 57, No. 11–12, 1776-1784
[19] Ćurković, P., Jerbić, B. & Stipančić, T. (2008) Hybridization of an adaptive genetic algorithm and ART 1 neural architecture for efficient path planning of a mobile robot, Trans of FAMENA, Vol. 32, 11-21
[20] Karafiotas, G., Smit, S. K. & Eiben, A. E. (2012) A generic approach to parameter control, Lecture Notes in Computer Science, 7248 LNCS, 366-375
[21] Ficici, S. G., Melnik, O. & Pollack, J.B. (2005) A game-theoretic and dynamical-systems analysis of selection methods in coevolution, IEEE Transactions on Evolutionary Computation, Vol. 9, No. 6, 580-602
[22] Kuzucuoglu, A. E. & Erdemir, G. (2011) Development of a Web-Based Control and Robotic Applications Laboratory for Control Engineering Education. Information Technology and Control, Kaunas, Technologija, Vol. 4, 352-358.
[23] Stipancic, T., Jerbic, B. & Curkovic, P. (2013) Bayesian approach to robot group control. Lecture Notes in Electrical Engineering. 130 LNEE, 109-119.