Multiobjective Dynamic Constrained Evolutionary Algorithm for Control of a Multi-Segment Articulated Manipulator

Krzysztof Michalak[1], Patryk Filipiak[2], and Piotr Lipinski[2]

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

[2] Computational Intelligence Research Group,
Institute of Computer Science,
University of Wroclaw, Wroclaw, Poland
{patryk.filipiak,piotr.lipinski}@cs.ii.uni.wroc.pl

Abstract. In this paper a multiobjective dynamic constrained evolutionary algorithm is proposed for control of a multi-segment articulated manipulator. The algorithm is tested in simulated dynamic environments with moving obstacles. The algorithm does not require previous training - a feasible sequence of movements is found and maintained based on a population of candidate movements. The population is evolved using typical evolutionary operators as well as several new ones that are dedicated for the manipulator control task. The algorithm is shown to handle manipulators with up to 100 segments. The increased maneuverability of the manipulator with 100 segments is well utilized by the algorithm. The results obtained for such manipulator are better than for the 10-segment one which is computationally easier to handle.

Keywords: inverse kinematics, multiobjective evolutionary optimization, constrained problems

1 Introduction

Inverse Kinematics (IK) is the problem of finding such configuration of an articulated robotic arm that satisfies certain constraints concerning the position and the orientation of its end effector. Applications of IK are very frequent in contemporary robotics, e.g. in steering of industrial planar robots [5], automatization of medical steerable needles [6], performing an optical motion capture [1] or robotic posture control and collision avoidance [8]. Although IK can be expressed in the algebraic form, it is highly inefficient to solve it explicitly. It was stated in [9] that finding the desired pose for the popular case of IK problem with 6 degrees of freedom is equivalent to solving a 16th order polynomial equation. In order to alleviate this difficulty, a number of numerical approaches were proposed instead.

2 Problem Statement

In this paper we consider a multi-segment articulated manipulator consisting of N_s segments that is mounted at a given point $O = [x_o, y_o]$. The segments are connected by joints $J_1, \ldots, J_{N_s}$ (the first joint being attached at the starting point O). The manipulator itself is described by a list of segment lengths $\{l_1, \ldots, l_{N_s}\}$. The position of the manipulator is determined by a list of relative angles $\{\alpha_1, \ldots, \alpha_{N_s}\}$, $\alpha_i \in (-\pi, \pi)$ (i.e. angles relative to previous segment orientation). In this paper we assume, that $\alpha_i = 0$ represents a segment pointing in the same direction as the previous one. The endpoint of the manipulator is expected to reach and remain as close as possible to a given target point $T = [x_t, y_t]$. The environment in which the manipulator operates includes a set of N_o obstacles $\{O_i\}_{i=1,\ldots,N_o}$. Each obstacle O_i is a convex polygon with M_i vertices. At no time t the manipulator may intersect any of the obstacles. Obviously, the manipulator has to move to reach the target point T and to avoid any obstacles. We assume that the movement of the manipulator is defined by setting values of angles between manipulator segments at discrete time instants. Therefore, the sequence of moves that the manipulator performs during all the N_t time steps of the entire simulation can be represented as: $\{\alpha_1(t), \ldots, \alpha_{N_s}(t)\}_{t=0,\ldots,N_t}$. A movement between time instants t and $t+1$ is performed as a linear change of all the angles: $\alpha_j(t+\delta) = \alpha_j(t) \cdot (1-\delta) + \alpha_j(t+1) \cdot \delta$, for $j = 1, \ldots, N_s$, $\delta \in [0, 1]$. At each time instant $t = 0, \ldots, N_{t-1}$ the algorithm has to calculate a set of angles $\{\alpha_1(t+1), \ldots, \alpha_{N_s}(t+1)\}$ for the time instant $t+1$ based on the current set of angles $\{\alpha_1(t), \ldots, \alpha_{N_s}(t)\}$ in such a way that the endpoint E of the manipulator remains possibly close to the target point T and the manipulator does not intersect any obstacles during the interval $[t, t+1]$.

3 Evolutionary Algorithm

At each time instant $t = 0, \ldots, N_{t-1}$ the evolutionary algorithm tries to find a new set of angles for the manipulator $\{\alpha_1(t+1), \ldots, \alpha_{N_s}(t+1)\}$ for the time instant $t+1$ based on the current set of angles $\{\alpha_1(t), \ldots, \alpha_{N_s}(t)\}$. The genotype of each specimen represents a candidate set of angles for the time instant $t+1$. The evolutionary algorithm proposed in this paper contains both some elements typical to evolutionary algorithms used for constrained optimization and some elements typical to dynamic optimization. For dealing with constraints the algorithm uses two mechanisms used in the Infeasibility Driven Evolutionary Algorithm (IDEA) [10]: a violation measure is used as one of the objectives and a fraction of the population is reserved for infeasible specimens. When solving dynamic optimization problems the loss of diversity is often an issue. To remedy this, random immigrants are added to every generation as proposed by [7].

Objectives and constraints

The algorithm minimizes three objectives f_1, f_2 and f_3 under two constraints g_1 and g_2. All objectives and constraints are calculated for N_δ simulation steps from time t to $t+1$. The n-th step ($n = 1, \ldots, N_\delta$) corresponds to time $t + \frac{n-1}{N_\delta - 1}$,

where: t - the time instant for which the evolutionary algorithm calculates the transition to $t+1$.

$\mathbf{f_1}$: The first objective is **the distance between the manipulator endpoint E and the target point T**. If there are no obstacles between E and T the Euclidean distance is used. If the $\overline{ET}$ line segment crosses an obstacle at points C_1 and C_2 the distance is calculated as: $f_1 = d_E(E,T) - d_E(C_1,C_2) + d_O$ where d_E denotes Euclidean distance and d_O is the shorter of two paths between C_1 and C_2 around the obstacle. The values for all simulation steps $n = 1, \ldots, N_\delta$ are averaged with weights equal to $\frac{n}{N_\delta}$. Therefore, a higher selective pressure is put on minimizing the distance from E to T at the end of the time interval $[t, t+1]$. This is intended to give the manipulator some freedom to adjust between time instants t and $t+1$, while promoting convergence to T towards the end of the time interval $[t, t+1]$.

$\mathbf{f_2}$: The second objective is **a measure of displacement of the manipulator between time instants t and $t+1$**: $f_2 = \sum_{k=1}^{N_s} \left[(x_{j_k}(t+1) - x_{j_k}(t))^2 + (y_{j_k}(t+1) - y_{j_k}(t))^2 \right]$, where: $x_{j_k}(t)$, $y_{j_k}(t)$ - the coordinates of the $k-th$ joint of the manipulator calculated for angles at the time instant t. Minimizing this objective is intended to limit the occurence of rapid or violent movements of the manipulator.

$\mathbf{f_3}$: The third objective is **a violation measure** proposed in [10] which represents how much the constraints are violated.

The constraints represent collisions with obstacles and self-intersections of the manipulator. Both g_1 and g_2 have to be 0 for the specimen to be feasible. Infeasible specimens have $g_1 > 0$ or $g_2 > 0$.

$\mathbf{g_1}$: The first constraint is **a measure of intersection with obstacles**. If the manipulator does not intersect with a given obstacle then the contribution of this obstacle to the g_1 constraint is 0. Otherwise, for each pair of intersection points C_1, C_2 the length of the shorter of the paths connecting C_1 and C_2 on the circumference of the obstacle is added to g_1. If the endpoint of the manipulator is inside the obstacle the length of the arm inside the obstacle is added.

$\mathbf{g_2}$: The second constraint is **a measure of self-intersections of the manipulator**. This measure is calculated as the sum of $\frac{1}{j \cdot k}$ for those j and k for which the manipulator segments $\overline{J_i J_{i+1}}$ and $\overline{J_k J_{k+1}}$ intersect.

Values of both constraints are summed for all simulations steps $n = 1, \ldots, N_\delta$.

Operators

The evolutionary algorithm proposed in this paper uses two genetic operators typically used for real-valued chromosomes: the SBX crossover described in [2] and the polynomial mutation operator introduced in [4]. Typical genetic operators mentioned above treat the set of angles between manipulator segments as just an array of real numbers. Additionally, three other operators are proposed in this paper that are dedicated for the task of articulated manipulator control.

Single joint mutation. If the polynomial mutation operator changes the value of an angle α_i positions of the segments that are placed after α_i change

significantly. This effect may cause a mutated manipulator position to become infeasible, especially in crowded environments. To mitigate this problem a second mutation operator was designed. The new operator uses the polynomial mutation operator to mutate individual angles. If an angle α_i is mutated to α'_i a correction is performed by calculating the change of the i-th angle $\delta_i = \alpha'_i - \alpha_i$ and by turning the joints that follow the mutated one in the opposite direciton: $\forall j > i : \alpha_j = \alpha_j - \delta_i$. This correction is intended to limit the influence of mutating one angle α_i on the entire part of the manipulator from joint J_i to the endpoint E.

The Unfold-3 operator. This operator is intended to help the manipulator straighten by replacing any three segments by two if possible. It is designed to be used for manipulators in which all the segments are equal to a given constant length L. By definition this operator is only applied when there exist two joints J_i and J_{i+3} for which $d_E(J_i, J_{i+3}) \leq 2L$, where $d_E(\cdot)$ is the Euclidean distance. The operator sets the angles α_i and α_{i+1} so that J_{i+2} and J_{i+3} are placed at the same positions as J_{i+3} and J_{i+4} were before the operator was applied. Then, all the angles that follow are corrected: $\forall i + 3 \leq j < N_s : \alpha_j = \alpha_{j+1}$. The last segment of the manipulator has no preceding position to which to refer, so the endpoint of the manipulator i directed towards the target point T.

The RepairSelfIntersections operator.

This operator tries to untangle the manipulator if self-intersections are present. First, a set of intersecting pairs of manipulator segments is identified:

$$I = \left\{ \langle j, k \rangle : \overline{J_j J_{j+1}} \text{ and } \overline{J_k J_{k+1}} \text{ intersect } \right\} \ . \tag{1}$$

Based on the set I the first and the last joint in the entangled part of the manipulator are identified:

$$j_f = min \left\{ j : \exists k : \langle j, k \rangle \in I \right\} + 1 \ , \tag{2}$$

$$j_l = max \left\{ k : \exists j : \langle j, k \rangle \in I \right\} \ . \tag{3}$$

One index j_{fix} is randomly selected from the range $j_f, \ldots, j_l$ with uniform probability. The angle $\alpha_{j_{fix}}$ is modified by setting $\alpha_{j_{fix}} = \eta \alpha_{j_{fix}}$, where η is a random number drawn from the $U[0, 1]$ distribution. This makes the segment $\overline{J_{j_{fix}} J_{j_{fix}+1}}$ closer to pointing straight with respect to the previous segment $\overline{J_{j_{fix}-1} J_{j_{fix}}}$.

3.1 The Main Loop

The main algorithm loop is presented in Algorithm 1. This main loop of the algorithm is executed for every time instant $t = 0, \ldots, N_t - 1$. The population P is initialized only once at the first time instant $t = 0$ before entering the main loop. The parameters that affect the execution of the algorithm are: N_{gen} - the number of generations, N_{pop} - population size, N_{rnd} - the number of random immigrants, N_f - the number of feasible specimens to select, and N_{inf} - number of infeasible specimens to select.

Algorithm 1 The main algorithm loop.

for $g = 1 \rightarrow N_{gen}$ **do**
  $P_{offspring} = \emptyset$
  $P_{mate} = \text{SelectMatingPool}(P)$
  for $i = 1 \rightarrow N_{pop}/2$ **do**
    $\langle O_1, O_2 \rangle = \text{Crossover}(P_{mate}[2 * i - 1],\ P_{mate}[2 * i])$
    $\text{Mutate}(O_1)$; $\text{Mutate}(O_2)$
    $\text{MutateOneJoint}(O_1)$; $\text{MutateOneJoint}(O_2)$
    $\text{RepairSelfIntersections}(O_1)$; $\text{RepairSelfIntersections}(O_2)$
    $P_{offspring} = P_{offspring} \cup \{O_1, O_2, \}$
  end for
  $P_{rnd} = \text{InitPopulation}(N_{rnd})$
  $\text{RepairSelfIntersections}(P_{rnd})$
  $P_{U3} = \text{Unfold-3}(P_{offspring}) \cup \text{Unfold-3}(P_{rnd})$
  $P_{av} = \text{AvoidObstacles}(P_{offspring}) \cup \text{AvoidObstacles}(P_{rnd}) \cup \text{AvoidObstacles}(P_{U3})$
  $\text{RepairSelfIntersections}(P_{av})$
  $\text{Evaluate}(P_{offspring} \cup P_{rnd} \cup P_{U3} \cup P_{av})$
  $P = P \cup P_{offspring} \cup P_{rnd} \cup P_{U3} \cup P_{av}$
  $\langle P_f, P_{inf} = Split(P)$
  $\text{Rank}(P_f)$; $\text{Rank}(P_{inf})$
  $P = P_{inf}[1 : N_{inf}] \cup P_f[1 : N_f]$
end for

Apart from the operators described in the "Operators" section the algorithm uses the following procedures.

SelectMatingPool Selects a mating pool using a binary tournament.

Crossover Generates offspring by applying the crossover operator with probability P_{cross} to a pair of parents or by copying them as they are with probability $1 - P_{cross}$.

InitPopulation Returns a given number of new specimens with randomly initialized genotypes. It is used to initialize a new population at time $t = 0$ and for generating random immigrants.

AvoidObstacles Modifies those candidate solutions that intersect obstacles. Moves the solution outside the obstacle by adjusting the last angle before the obstacle, so that the manipulator segment becomes tangent to an ϵ-envelope of the obstacle.

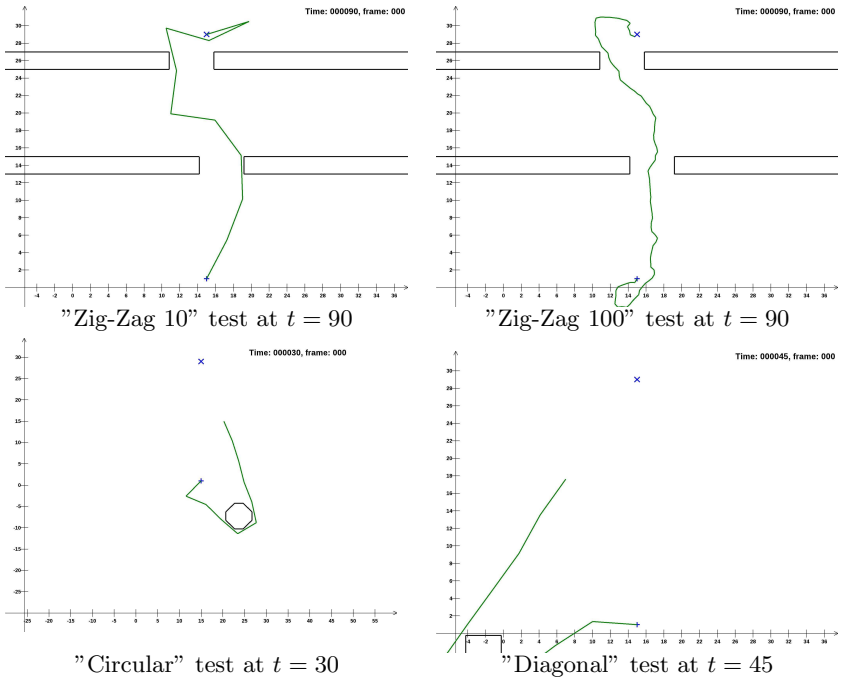
Evaluate Calculates values of the f_1, f_2 and f_3 objectives as well as the g_1 and g_2 constraints.

Split Splits the population to feasible and infeasible specimens (those having $g_1 = 0$ and $g_2 = 0$ and those having $g_1 > 0$ or $g_2 > 0$ respectively).

Rank Ranks a set of specimens using nondominated sorting and then crowding distance sorting used in the NSGA-II algorithm [3]. The set of feasible specimens P_f and the set of infeasible specimens P_{inf} are ranked separately.

4 Experiments and Results

The proposed algorithm was tested on four scenarios involving different obstacle courses with different movement types and different number of manipulator segments. In all tests the number of time instants was $N_t = 100$. The first three scenarios involved manipulators with $N_s = 10$ segments while the last scenario involved a manipulator with $N_s = 100$ segments. All the scenarios were tested for the number of generations $N_{gen} = 50, 100, 200$ and population sizes $N_{pop} = 50, 100, 200$. For each pair of parameters 10 iterations were performed. Visualizations of these scenarios are given in Figure 1.



"Zig-Zag 10" test at $t = 90$ "Zig-Zag 100" test at $t = 90$

"Circular" test at $t = 30$ "Diagonal" test at $t = 45$

Fig. 1. Obstacle courses used in the experiments

The parameters of the evolutionary algorithm were set as follows: crossover probability $P_{cross} = 0.9$, crossover distribution index $\eta_{cross} = 15$, mutation probability $P_{mut} = 0.1$, mutation distribution index $\eta_{mut} = 20$, percentage of random immigrants $N_{rnd}/N_{pop} = 20\%$, percentage of infeasible solutions $N_{inf}/N_{pop} = 20\%$. The effectiveness of the algorithm in keeping the manipulator endpoint E close to the target point T was measured by calculating the average Euclidean distance $d_E(E, T)$ and the fraction q of time instants during which $d_E(E, T) < 0.5$. In Table 1 the average Euclidean distance $d_E(E, T)$ be-

tween the manipulator endpoint E and the target point T and the fraction q of time instants during which $d_E(E,T) < 0.5$ are given.

Table 1. The average Euclidean distance $d_E(E,T)$ between the manipulator endpoint E and the target point T and the fraction q of time instants during which $d_E(E,T) < 0.5$ obtained in the tests

Circular		$N_{pop}=50$		$N_{pop}=100$		$N_{pop}=200$	
		$d_E(E,T)$	q	$d_E(E,T)$	q	$d_E(E,T)$	q
N_{gen}	50	10.64	28.69%	10.55	27.44%	10.97	26%
	100	9.43	34.46%	10.67	28.37%	9.74	32.09%
	200	9.96	29.69%	9.32	33.15%	8.41	32.53%

Diagonal		$N_{pop}=50$		$N_{pop}=100$		$N_{pop}=200$	
		$d_E(E,T)$	q	$d_E(E,T)$	q	$d_E(E,T)$	q
N_{gen}	50	5.47	57.76%	4.57	60.98%	4.33	62.02%
	100	4.84	58.34%	4.50	61.16%	4.19	62.44%
	200	4.88	56.44%	4.38	61.57%	4.23	61.55%

Zig-Zag 10		$N_{pop}=50$		$N_{pop}=100$		$N_{pop}=200$	
		$d_E(E,T)$	q	$d_E(E,T)$	q	$d_E(E,T)$	q
N_{gen}	50	5.00	16.6%	3.95	28.59%	3.42	33.35%
	100	4.44	24.93%	3.32	38.19%	2.61	48.14%
	200	2.91	44.23%	2.94	41.94%	2.30	54.46%

Zig-Zag 100		$N_{pop}=50$		$N_{pop}=100$		$N_{pop}=200$	
		$d_E(E,T)$	q	$d_E(E,T)$	q	$d_E(E,T)$	q
N_{gen}	50	4.40	24.53%	4.01	29.98%	3.04	38.87%
	100	4.74	31.26%	2.87	42.84%	2.29	52.07%
	200	3.12	44.30%	2.82	48.05%	1.78	61.03%

Comparison of values from Table 1 shows that the manipulator with 100 segments is handled effectively by the proposed algorithm. The average distance from the target $d_E(E,T)$ obtained for the "Zig-Zag 100" test is very similar to that obtained for the "Zig-Zag 10" test. The fraction q of time instants during which $d_E(E,T) < 0.5$ is higher for the "Zig-Zag 100" test.

In two problems "Circular" and "Diagonal" it might be beneficial for the algorithm to backtrack and "go around" the obstacle. For these problems increasing the values of the parameters (the population size N_{pop} and the number of generations N_{gen}) provides only a moderate improvement in solution quality. This may suggest that the algorithm should include elements of planning. For example it could be beneficial to optimize a sequence of moves for several time instants, not just one movement at a time. Increasing the values of the N_{pop} and N_{gen} parameters improves the results significantly in the "Zig-Zag 10" and "Zig-Zag 100" tests which feature more obstacles and a narrower path to the

target. This effect may be caused by a large number of specimens required to find a feasible movement in such crowded space.

5 Conclusions

In this paper a multiobjective dynamic constrained evolutionary algorithm was proposed for control of a multi-segment articulated manipulator. The main advantage of the presented algorithm is that it does not require any tuning for the manipulator parameters nor for any particular environment. The test with 100-segment manipulator shows, that the algorithm is able to utilize the increased maneuverability of this manipulator compared to the one with 10 segments. In an environment with moving obstacles the algorithm managed to keep the endpoint of the 100-segment manipulator closer on average to the target point T than in the case of the 10-segment one and for a larger fraction of time.

Further work may include an elaboration of a similar method for 3D space. This would require, at the very least, defining the distance around an obstacle in a way applicable to 3D and a modification of evaluation of constraint g_1 (intersections with obstacles). From the point of view of development of intelligent methods it may be useful to employ AI planning which would allow going around obstacles in directions opposite to where the target point is.

References

1. Aristidou, A., Lasenby, J.: Motion capture with constrained inverse kinematics for real-time hand tracking. In: Proceedings of the IEEE International Symposium on Communications, Control and Signal Processing (ISCCSP 20120). pp. 1–5 (2010)
2. Deb, K., Agarwal, R.: Simulated binary crossover for continuous search space. Complex Systems 9(2), 115–148 (1995)
3. Deb, K., et al.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197 (2002)
4. Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design. Computer Science and Informatics 26, 30–45 (1996)
5. Dong, H., et al.: Workspace density and inverse kinematics for planar serial revolute manipulators. Mechanism and Machine Theory 70, 508–522 (2013)
6. Duindam, V., Xu, J., Alterovitz, R., Sastry, S., Goldberg, K.: Three-dimensional motion planning algorithms for steerable needles using inverse kinematics. The International Journal of Robotics Research 29(7), 789–800 (2010)
7. Grefenstette, J.: Genetic algorithms for changing environments. In: Parallel Problem Solving from Nature 2. pp. 137–144. Elsevier (1992)
8. Kallmann, M.: Analytical inverse kinematics with body posture control. Computer Animation and Virtual Worlds 19(2), 79–91 (2008)
9. Pieper, D., L.: The kinematics of robots under computer control. Phd thesis, Stanford University (1968)
10. Singh, H.K., Isaacs, A., Ray, T., Smith, W.: Infeasibility driven evolutionary algorithm (IDEA) for engineering design optimization. In: Wobcke, W., Zhang, M. (eds.) AI 2008: Advances in Artificial Intelligence, 21st Australasian Joint Conference on Artificial Intelligence, Auckland, New Zealand, December 1-5, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5360, pp. 104–115 (2008)