A Predictive Evolutionary Algorithm for Dynamic Constrained Inverse Kinematics Problems

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

[1]Institute of Computer Science,
University of Wroclaw, Wroclaw, Poland
lipinski@ii.uni.wroc.pl
[2]Institute of Business Informatics
Wroclaw University of Economics, Wroclaw, Poland

Abstract. This paper presents an evolutionary approach to the Inverse Kinematics problem. The Inverse Kinematics problem concerns finding the placement of a manipulator that satisfies certain conditions. In this paper apart from reaching the target point the manipulator is required to avoid a number of obstacles. The problem which we tackle is dynamic: the obstacles and the target point may be moving which necessitates the continuous update of the solution. The evolutionary algorithm used for this task is a modification of the Infeasibility Driven Evolutionary Algorithm (IDEA) augmented with a prediction mechanism based on the ARIMA model.

1 Introduction

The recent developments in robotics, cybernetics or even computer graphics have led to a growing interest in the Inverse Kinematics problem, which concerns finding values of parameters describing some object of skeletal structure (e.g. a robot or virtual model of human body) which allow an intended pose of this object to be achieved. The Inverse Kinematics approach is usually employed when it is more important to find an acceptable placement of some manipulator (e.g. robotic arm) rather than provide an actual control for it. Thus, this kind of problem arise mainly in the area of visualization (computer graphics, animation, etc.) and as a sub-problem called *arm configuration selection* in the wider context of reaching the subtask in manipulation planning problems [3, 18].

One of the simplest formulations of the Inverse Kinematics problem is as follows: find such a pose of a manipulator that, given a starting position, a desired position of its ending is achieved. This simple problem formulation may be modified in several ways. The manipulator can have a more complicated geometry than a simple chain of linear segments. For example a tree–like structure can be used to represent a hand with the movement of each finger simulated individually. It is also possible, that segments of the manipulator are described by a different set of parameters, for example segments may bend or stretch. Another modification is the introduction of obstacles. Shape of the obstacles may vary from simple blocks to sophisticated labyrinths.

** corresponding author

Although the classic Inverse Kinematics problem can be solved by some classic analytical methods, more advanced version of the problem, e.g. with additional constraints or obstacles, require more efficient algorithms. There are a number of evolutionary algorithms applied to various versions of the classic Inverse Kinematics problem.

In this paper we use the evolutionary approach in order to solve an Inverse Kinematics problem with an articulated arm and dynamic obstacles and target point. A modification of the Infeasibility Driven Evolutionary Algorithm (IDEA) [15, 16] seems to be particularly useful for this problem, because many solutions which collide with obstacles while infeasible in physical world may constitute useful approximations of a feasible solution. Because of the assumption that obstacles and the goal are dynamic we employ a prediction mechanism based on the ARIMA model [4] to improve algorithm performance.

This paper is structured as follows: in Section 2 a formal problem definition is given. Section 3 presents some classical approaches to various Inverse Kinematics problems. Section 4 introduces a detailed description of the proposed evolutionary algorithm with a prediction mechanism. In section 5 the experimental setup is described and the results of experiments are presented. Section 6 concludes the paper.

2 Problem definition

There is a broad range of Inverse Kinematics problems with their own slightly different formal definitions.

In this paper we focus on a problem with an articulated manipulator (like robotic arm) in the shape of a chain of any number M of fixed–length linear segments. The problem is to find angles at all joints of such a manipulator given a starting position and a desired position of its ending. The manipulator has one fixed starting point S and its shape is determined by a set of line segment lengths $l_1, \ldots, l_M$. The objective is to reach a target point O. Given the fixed coordinates of the starting point S the position of the manipulator can be described by a set of numbers $\alpha_1, \ldots, \alpha_M$ which represent angles at which manipulator segments are positioned. There are two possible interpretations of these numbers: as absolute or as relative angles. In this paper we treat the numbers as relative angles, so the actual angle of $k-th$ manipulator segment is $\sum_0^k \alpha_k$, where α_0 is an arbitrarily chosen initial angle (we chose $\alpha_0 = \pi/2$ - the direction straight along Y axis). In the environment in which the manipulator operates a number of obstacles are introduced.

The problem that we approach in this paper is dynamic which means that the obstacles move and the target point may also change its location. The motion of the obstacles may be linear or circular with varying values of velocity and period.

3 Classical approach

There exists a broad range of solutions that address many variants of the Inverse Kinematics problem upon which the Cyclic Coordinate Descent (CCD) [7] and the Jacobian–based methods [2, 19] are considered nearly canonical. Performance of CCD, though, is limited by the fact that it optimizes each joint separately. What is more, neither of the two methods allows for incorporating constraints.

Some numerical methods were introduced in [9, 10] however they tend to fall into local optima in more complex tasks. Universal and rapid solutions that can address various static constrained Inverse Kinematics problems using Genetic Algorithms and Niching Techniques were also presented in [12, 13, 17].

One ubiquitous disadvantage of many evolutionary methods lies in their low reactivity to changes in dynamic environment. On the other hand, analytical methods can typically be applied only to special cases of Inverse Kinematics and strictly depend on precise analytical model that is usually difficult to construct in real–life problems.

Many recent developments in artificial intelligence systems reveal that applying hybrid technics gives promising results in difficult computational problems [5, 6, 8] by employing various additional tools, e.g. learning machines [1], neural networks [14], and/or statistical methods (as it is proposed in this paper).

4 Evolutionary algorithm with prediction

The algorithm used in this paper is based on the Infeasibility Driven Evolutionary Algorithm (IDEA) [15, 16] that is well–fitted to some dynamic problems. IDEA increases the reactivity to changes by maintaining in population a fraction of infeasible individuals with the highest fitness value. Let P_{Size} be the size of a population and $0 < \alpha \ll 1$ be the rate of infeasible solutions which means there are $\lfloor \alpha P_{Size} \rfloor$ infeasible individuals kept in population.

An evaluation function and constraint functions (i.e. functions counting the number of crossings with obstacles) are tested for changes at each t-th time step where $t = 2, \ldots, N_G$. If any such change is detected, a whole population P_{t-1} is re-evaluated.

Evolutionary operators are enclosed in the Sub-evolve function which comprises of $N_{G'}$ iterations of crossover and mutation. Being invoked with an argument P_{t-1} as an initial population Sub-evolve function returns the resulting population C_{t-1} of sub-evolution unioned with P_{t-1} then reduced to the fractions of $\lceil (1-\alpha) P_{Size} \rceil$ feasible and $\lfloor \alpha P_{Size} \rfloor$ infeasible individuals both with the highest fitness value.

Algorithm 1 presents the evolutionary framework using IDEA for dynamic optimization problems proposed in [16].

4.1 Prediction mechanism

The modification suggested in this paper integrates original IDEA with a prediction model in the following manner. In each iteration $t \geq H_{Start}$ an additional hidden population H_{t+1} of the size P_{Size} is initialized randomly then evaluated, sub-evolved and reduced as described above. However, instead of proper values of an evaluation function at the current time step t predictions of corresponding values at $t+1$ are used. Later on, at the beginning of the next iteration, an additional hidden population H_t is re-evaluated according to the actual evaluation function that is known at the moment. After that, the population P_{t-1} is unioned with H_t and reduced back to P_{Size} individuals in the same manner as described above, i.e. including $\lfloor \alpha P_{Size} \rfloor$ best infeasible solutions.

Algorithm 1 Evolutionary framework for Inverse Kinematics problem where P_{Size} = population size, N_{Gen} = number of generations and H_{Start} is the number of iteration at which the prediction begins.

```
$P_1$ = InitPopulation()
Evaluate($P_1$)
**for** $t = 2 \rightarrow N_{Gen}$ **do**
    **if** the function has changed **then**
        Evaluate($P_{t-1}$)
    **end if**
    $C_{t-1}$ = Sub-evolve($P_{t-1}$)
    Evaluate($C_{t-1}$)
    $P_t$ = Reduce($P_{t-1} + C_{t-1}$)
**end for**
```

As a prediction model ARIMA(p, d, q) [4] is used in order to forecast changes in the location of obstacles and/or target point because this is the factor that implicitly influence the values of evalution function. Previous positions of such dynamic objects of environment are collected in time series. Their new current coordinates are stored each time a change of location is detected. As a consequence, a predicted state of environment for time step $t + 1$ can be estimated at time step t then used for the evaluation of a hidden population.

Algorithm 2 Evolutionary framework for Inverse Kinematics problem with a prediction model where P_{Size} = population size, N_{Gen} = number of generations and H_{Start} is the number of iteration at which the prediction begins.

```
$P_1$ = InitPopulation()
Evaluate($P_1$)
**for** $t = 2 \rightarrow N_{Gen}$ **do**
    **if** the function has changed **then**
        Evaluate($P_{t-1}$)
        **if** $t - 1 \geq H_{Start}$ **then**
            Evaluate($H_t$)
            $P_{t-1}$ = Reduce($P_{t-1} + H_t$)
        **end if**
    **end if**
    $C_{t-1}$ = Sub-evolve($P_{t-1}$)
    Evaluate($C_{t-1}$)
    $P_t$ = Reduce($P_{t-1} + C_{t-1}$)
    **if** $t \geq H_{Start}$ **then**
        $H_t$ = InitPopulation()
        PredictionEvaluate($H_t$)
        $D_t$ = PredictionSub-evolve($H_t$)
        $H_{t+1}$ = Reduce($H_t + D_t$)
    **end if**
**end for**
```

Algorithm 2 presents the framework of the described process in a form of pseudocode. For simplicity, the proposed algorithm with ARIMA prediction model will be referred to as IDEA-ARIMA for the remainder of this paper.

4.2 Evaluation function

Each individual in populations P_t or H_t is represented by the following sequence of angles

$$(\alpha_1, \ldots, \alpha_M) \in [A_1, B_1] \times [A_2, B_2] \times \ldots \times [A_M, B_M], \tag{1}$$

where $0 \leq A_i < B_i \leq 2\pi$ determines the lower and the upper bounds (respectively) of the angle range of i-th manipulator's segment for $i = 1, \ldots, M$.

The evaluation of an individual is equal to Euclidean distance between the end of the last (i.e. M-th) segment of a manipulator and the target point O providing there are no obstacles crossed by the line between them.

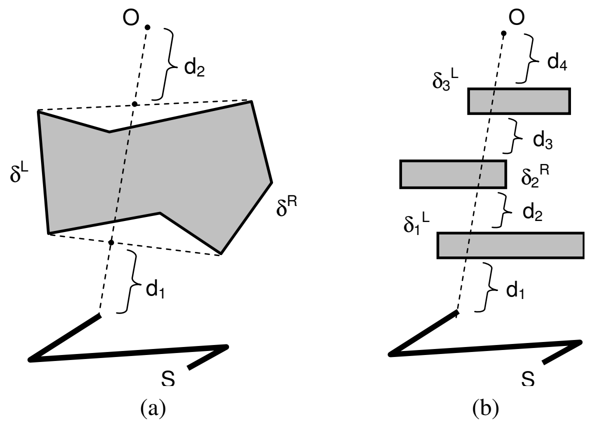


Fig. 1. Evaluation of an individual in the case of (a) one obstacle ahead, (b) three obstacles ahead.

If there is one obstacle crossed by this line, the convex hull of the obstacle is computed. In this case, two ways to pass by the obstacle emerge, i.e. by following the clockwise (δ^R) or the counterclockwise direction (δ^L). As a result, the shorter path of these two is selected. In the example depicted in Figure 1(a) the distance between the end–effector and the target point is split into two segments of the lengths d_1, d_2 and two possible paths of the lengths δ^L, δ^R. The evaluation equals $d_1 + \delta^L + d_2$, because clearly $\delta^L < \delta^R$.

In the case when more than one obstacle is crossed by the mentioned line, all obstacles are sorted according to distances between them and the end of the last segment of a manipulator. Then, consecutive obstacles are treated separately as described above. Figure 1(b) presents such case. Here, the distance is split into segments of the lengths d_1, d_2, d_3, d_4 and paths of the lengths $\delta_1^L, \delta_1^R, \delta_2^L, \delta_2^R, \delta_3^L, \delta_3^R$ hence the evaluation is

$d_1 + \delta_1^L + d_2 + \delta_2^R + d_3 + \delta_3^L + d_4$. Note that the evaluation function defined in this manner rewards passing by the obstacles rather than greedily searching for the shortest Euclidean distance to the target as it would completely ignore the presence of obstacles.

Individuals coding manipulators that cross an obstacle in at least one point are considered infeasible. A number of such crossings defines the order among infeasible solutions, i.e. the greater number of crossings, the more infeasible the solution.

5 Experiments

The proposed algorithm was tested on three types of benchmark problems described below. In each problem, the aim was to find a pose of manipulator that minimizes the distance (measured in the manner that was presented in Section 4) between its last segment and the target point $O \in \mathbb{R}^2$ without crossing any obstacles. For simplicity, all feasible ranges of angles α_i ($i = 1, \dots, M$) were set to $[0, 2\pi]$.

Obstacles and/or target point were able to change their locations in time. For any given point $(x, y) \in \mathbb{R}^2$ of such dynamic elements the position of it at time step t, namely (x_t, y_t), was calculated using the following predefined motion function

$$\mathbb{R}^2 \times \{1, \dots, \tau\} \ni t \longmapsto \phi(x, y, t) = (x_t, y_t). \tag{2}$$

Crossover and mutation probability were set to 0.9 and 0.1 respectively in both IDEA and IDEA-ARIMA. The fraction of infeasible yet promising individuals formed 20% of the population at all time steps.

Parameters of ARIMA prediction model were set to (1, 1, 1) as it resulted in optimal forecasting accuracy for examined problems. First H_{Start} = 16 iterations of any run of IDEA-ARIMA were dedicated only for collecting information about locations of obstacles for the sake of further predictions. These initial iterations were not taken into account in either of statistics presented in this section since observable results of both algorithms do not differ within this time period.

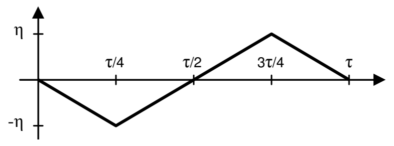
5.1 Benchmark I

Shape and alignment of obstacles including the way their locations change in time is depicted in Figure 2. Point $S \in \mathbb{R}^2$ represents the fixed location of a manipulator. The distance d between S and the target point O was set to 4. The manipulator itself was built out of $M = 4$ joints of the length $l = 2$. The test procedure lasted for $\tau = 16$ iterations (excluding H_{Start}) during which the three obstacles (marked as light-gray rectangles in Figure 2) where moving from left to right and vice versa in one of the two following manners:

(a) *uniform* – with the motion function $\phi(x, y, t) = (x + \omega(t), y)$ where $(x, y) \in \mathbb{R}^2$ and $\omega(t)$ defined as follows

$$\omega(t) = \begin{cases} -4\eta t/\tau & t \in [0, \tau/4) \cap \mathbb{Z}, \\ 4\eta t/\tau - 2\eta & t \in [\tau/4, 3\tau/4) \cap \mathbb{Z}, \\ -4\eta t/\tau + 4\eta & t \in [3\tau/4, \tau] \cap \mathbb{Z} \end{cases} \tag{3}$$

for a given amplitude $\eta > 0$. It is worth mentioning that such $\omega(.)$ is a discritization of the function depicted in the following graph



(b) *sinusoidal* – with the motion function defined as

$$\phi(x, y, t) = \left(x + \eta \sin\left(\frac{2\pi t}{\tau}\right), y\right), \tag{4}$$

where $(x, y) \in \mathbb{R}^2,\ t = 1, \ldots, \tau,\ \eta > 0$.

Table 1 presents the summary of results for 10 runs of IDEA and IDEA-ARIMA with population sizes (*pop*) 20, 50, 100 and 10, 20 sub-EA steps (*sub*) and motion amplitude η = 2.5. For each combination of *pop* and *sub* the three following values are presented:

1. *mean* – mean value of best feasible solutions found by given algorithm at each time step $t = 1, \ldots, 10 \cdot \tau$,
2. *wins* – percent of time steps t when the best feasible solution found by given algorithm was better than the corresponding one found by the latter algorithm,
3. $\Delta < 0.5$ – percent of time steps t when the best feasible solution found by given algorithm was worse than the corresponding one found by the latter algorithm but not worse than 0.5 (i.e. solution that is still satisfying one though not the best found).

Note that IDEA *wins* and IDEA-ARIMA *wins* do not always sum up to 100% (especially in cases where population size was relatively low). It is because none of the algorithms was able to find a feasible solution in some iterations.

As it is clearly seen, IDEA-ARIMA outperforms IDEA in all presented cases. The Student's t-test performed on each combination of population size and sub-EA steps used within the experiment revealed the superiority of IDEA-ARIMA at the significance level $\alpha < 0.01$. Additionally, values in column $\Delta < 0.5$ (fraction of still satisfying but not the best solutions found) show that the number of remaining time steps when IDEA-ARIMA performs rather poorly is relatively small.

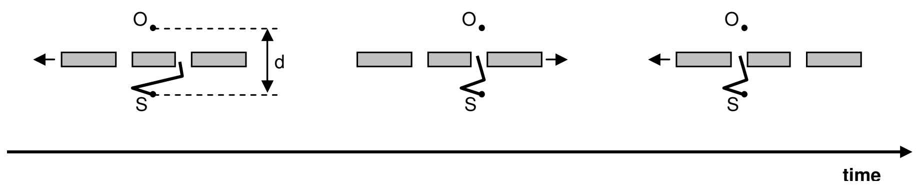


Fig. 2. Illustration of the movement of obstacles in Benchmark I. On each frame point S represents the location of a manipulator while O is the target point. The distance d between these points was set to 4.

uniform motion function

		IDEA			IDEA-ARIMA		
pop	sub	mean	wins	$\Delta < 0.5$	mean	wins	$\Delta < 0.5$
20	10	7.0954	28%	13%	1.4139	71%	17%
50	10	0.7419	29%	23%	0.3128	71%	21%
100	10	0.4841	29%	51%	0.1745	71%	24%
20	20	1.6240	38%	30%	0.4937	61%	23%
50	20	1.1384	30%	44%	0.1729	68%	23%
100	20	0.2537	24%	60%	0.0727	76%	21%

sinusoidal motion function

		IDEA			IDEA-ARIMA		
pop	sub	mean	wins	$\Delta < 0.5$	mean	wins	$\Delta < 0.5$
20	10	1.6738	34%	13%	0.5580	64%	12%
50	10	1.1252	31%	27%	0.1586	66%	18%
100	10	0.9452	30%	32%	0.0565	70%	26%
20	20	1.4262	34%	25%	0.3199	65%	16%
50	20	1.0245	28%	40%	0.0649	73%	20%
100	20	1.3396	15%	51%	0.0156	85%	14%

Table 1. Summary of results for Benchmark I after 10 runs of IDEA and IDEA-ARIMA with population sizes 20, 50, 100 and 10, 20 sub-EA steps.

It is visible in Table 1 (as it was expected) that effectiveness of both evolutionary algorithms usually grows with the increase of population size and number of iterations. On the other hand, the dynamic character of the presented benchmark problem reveals that there are time steps when the target point is quite easy to reach by the manipulator and other ones when obstacles are extremely hard to avoid. This is why it was impossible to cross the certain level of *wins* despite increasing parameters of algorithm.

		IDEA			IDEA-ARIMA		
pop	sub	mean	wins	$\Delta < 0.5$	mean	wins	$\Delta < 0.5$
20	10	3.1932	23%	26%	2.2538	77%	12%
50	10	2.9021	23%	20%	1.3467	77%	21%
100	10	2.9346	17%	16%	1.1059	83%	10%
20	20	2.4268	49%	21%	2.3896	51%	21%
50	20	2.1339	34%	34%	1.6165	66%	20%
100	20	2.1130	23%	25%	0.7715	77%	18%

Table 2. Summary of results for Benchmark II after 10 runs of IDEA and IDEA-ARIMA with population sizes 20, 50, 100 and 10, 20 sub-EA steps.

5.2 Benchmark II

Figure 3 depicts the dynamics of Benchmark II. Two vertices of obstacle change their locations by moving outwards for the first $\tau/2$ time steps and inwards for another $\tau/2$ time steps as the arrows indicate.

The parameters were set as follows: number of joints $M = 7$, length of joints $l = 2$, number of time steps $\tau = 16$, distance to target point $d = 9$ and sinusoidal motion amplitude $\eta = 1$.

The use of evaluation function described in Section 4 plays an important role in this benchmark problem because the obstacle is a concave figure. The shape of the obstacle is designed to have a local optimum very near point O in the terms of Euclidean distance.

Table 2 summarizes the results of 10 runs of both algorithms. IDEA-ARIMA clearly outperforms IDEA. The Student's t-test revealed the statistical significance (at the level $\alpha < 0.01$) in all cases except for $pop = 20, sub = 20$. Apparently, it is caused by the higher level of difficulty of this problem.

5.3 Benchmark III

Unlike Benchmarks I and II where obstacles were moving and target point O was static, in this case both obstacles and target point are circulating around point $S = (x_S, y_S) \in \mathbb{R}^2$ where manipulator is located. Let for all $(x, y) \in \mathbb{R}^2$ and $t = 1, \ldots, \tau$ motion function be defined as $\phi(x, y, t) = (x_t, y_t)$ where

$$x_t = (x - x_S)\cos\beta(t) + (y - y_S)\sin\beta(t) + x_S, \quad (5)$$

$$y_t = (x - x_S)\sin\beta(t) + (y - y_S)\cos\beta(t) + y_S. \quad (6)$$

Two variants of $\beta(t)$ defined in time domain determine two motion functions considered in this benchmark problem:

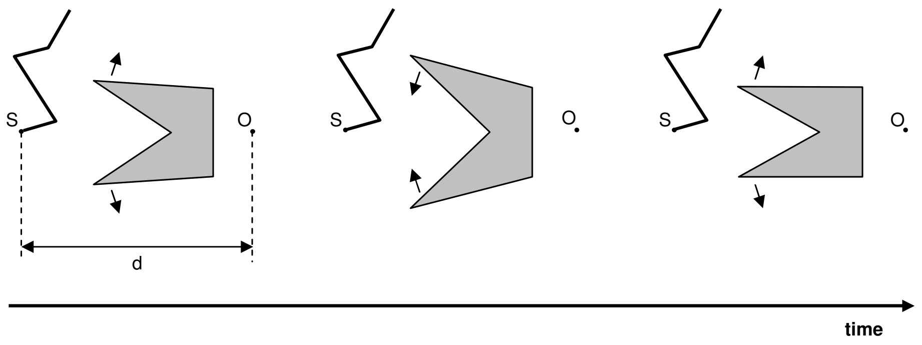


Fig. 3. Illustration of the movement of obstacles in Benchmark II. On each frame point S represents the location of a manipulator while O is the target point. The distance d between these points was set to 9.

(a) for *uniform-circular* motion function defined as

$$\beta(t) = \frac{2\pi t}{\tau}, \qquad t = 1, \dots, \tau, \tag{7}$$

(b) for *sigmoid-circular* motion function defined as

$$\beta(t) = \frac{2\pi}{1 + \exp(-6 + \frac{12t}{\tau})}, \qquad t = 1, \dots, \tau. \tag{8}$$

Figure 4 presents the environment (i.e. shape and alignment of obstacles, location of manipulator and target point) and illustrates sigmoid-circular motion of objects.

Table 3 summarizes the results of 10 runs of IDEA and IDEA-ARIMA with parameters: number of joints $M = 4$, length of joints $l = 2$, number of time steps $\tau = 32$, distance to target point $d = 5$. The use of IDEA gave satisfactory results in less then 60% of time steps. IDEA-ARIMA noted *wins* around twice as often than IDEA and produced satisfactory results in almost all time steps in this benchmark. The Student's t-test revealed the statistical significance (at the level $\alpha < 0.01$) in all examined cases.

6 Conclusions

In this paper an extension of the Infeasibility Driven Evolutionary Algorithm (IDEA) with a prediction mechanism based on ARIMA model is proposed for solving dynamic constrained Inverse Kinematics problem of reaching target point with M-segmented manipulator.

Experiments performed on a few dynamic benchmark problems revealed that IDEA-ARIMA outperforms IDEA in all examined cases by producing satisfactory results in almost all time steps, which proves that the prediction of the dynamic objective function may lead to better results. However, further studies are necessary to examine the relevance of prediction in more random environments.

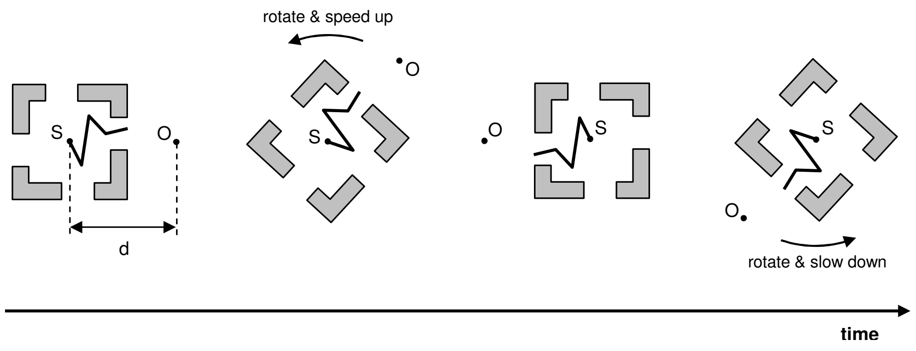


Fig. 4. Illustration of sigmoid-circular movement of obstacles in Benchmark III. On each frame point S represents the location of a manipulator while O is the target point. The distance d between these points was set to 5.

uniform-circular motion function

		IDEA			IDEA-ARIMA		
pop	sub	mean	wins	$\Delta < 0.5$	mean	wins	$\Delta < 0.5$
20	10	1.9126	26%	9%	0.9376	67%	8%
50	10	1.0658	29%	11%	0.4296	68%	11%
100	10	0.6066	28%	20%	0.1836	71%	8%
20	20	1.0224	31%	31%	0.5842	67%	15%
50	20	0.6549	25%	28%	0.1201	69%	19%
100	20	0.8457	23%	48%	0.0591	71%	20%

sigmoid-circular motion function

		IDEA			IDEA-ARIMA		
pop	sub	mean	wins	$\Delta < 0.5$	mean	wins	$\Delta < 0.5$
20	10	1.6738	18%	18%	0.5580	67%	12%
50	10	1.1252	23%	30%	0.1586	73%	20%
100	10	0.9452	25%	36%	0.0565	70%	24%
20	20	1.4262	26%	22%	0.3199	57%	20%
50	20	1.0245	20%	37%	0.0649	71%	18%
100	20	1.3396	24%	32%	0.0156	70%	23%

Table 3. Summary of results for Benchmark III after 10 runs of IDEA and IDEA-ARIMA with population sizes 20, 50, 100 and 10, 20 sub-EA steps.

This paper shows that ARIMA–based prediction delivers promising results when applied to some dynamic constrained Inverse Kinematics problems that can occur in real–life robotics applications.

References

1. Abraham, A., Corchado, E., Corchado, J., M., *Hybrid learning machines.* Neurocomputing, Vol. 72(13-15), pp. 2729-2730, 2009.
2. Balestrino, A., Maria, G. De, Sciavicco, L., *Robust control of robotic manipulators*, [in] Proceedings of the 9th IFAC World Congress, Vol. 5, pp. 2435-2440, 1984.
3. Bertram, D., Kuffner, J., Dillmann, R., Asfour, T., *An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators*, [in] Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1874-1879, 2006.
4. Box, G., E., P., Jenkins, G., M., *Time series analysis: Forecasting and control*, Revised edition, Holden-Day, San Francisco, 1976.
5. Corchado, E., Abraham A., Carvalho, A, *Hybrid intelligent algorithms and applications.* Information Sciences, Vol. 180(14), pp. 2633-2634, 2010.
6. Corchado, E., Graa, M., Wozniak, M., *New trends and applications on hybrid artificial intelligence systems. Neurocomputing*, Vol 75, Issue 1, pp. 61-63, 2012.
7. Fêdor, Martin, *Application of inverse kinematics for skeleton manipulation in real-time*, [in] Proceedings of the 19th spring conference on Computer graphics, ACM, pp. 203-212, 2003.
8. Garca, S., Fernndez, A., Luengo, J., Herrera F., *Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power.* Information Sciences, Vol. 180(10), pp. 2044-2064, 2010.

9. Goldenberg, A., A., Benhabib, B., Fenton, G. *A complete generalized solution to the inverse kinematics of robots*, IEEE Journal of Robotics and Automation, Vol. RA-1, No. 1, 1985.
10. Goldenberg, A., A., Lawrence, D., L., *A generalized solution to the inverse kinematics of robot manipulators*, ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 107, pp. 103-106, 1985.
11. Hatzakis, I., Wallace, D., *Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach*, [in] Proceedings of the GECCO 2006, ACM, pp.1201-1208, 2006.
12. Karla, P., Mahapatra, P., B., Aggarwal, D., K., *On the solution of multimodal robot inverse kinematics function using real-coded genetic algorithms*, [in] Proceedings of IEEE Internation Conference on Systems, Man and Cybernetics, Vol. 2, pp. 1840-1445, 2003.
13. Karla, P., Mahapatra, P., B., Aggarwal, D., K., *On the comparison of niching strategies for finding the solution of multimodal robot inverse kinematics*, [in] Proceedings of IEEE Internation Conference on Systems, Man and Cybernetics, Vol. 6, pp. 5356-5361, 2004.
14. Pedrycz, W., Aliev, R., *Logic-oriented neural networks for fuzzy neurocomputing*. Neurocomputing, Vol. 73, Issues 1-3, pp. 10-23, 2009.
15. Singh, H., K., Isaacs, A., Tapabrata, R.,*Infeasibility Driven Evolutionary Algorithm (IDEA) for engineering design optimization*, in Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence, pp. 104-115, 2008.
16. Singh, H., K., Isaacs, A., Nguyen, T., T., Ray, T., Yao, X., *Performance of infeasibility driven evolutionary algorithm (IDEA) on constrained dynamic single objective optimization problems*, [in] Proceedings of IEEE Congress on Evolutionary Computation, CEC '09, pp. 3127-3134, 2009.
17. Tabandeh, S., Clark, C., Melek, W., *A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering*, [in] Proceedings of IEEE Congress on Evolutionary Computation, CEC 2006, pp. 1815-1822, 2006.
18. Wilfong, G., *Motion planning in the presence of movable obstacles*, [in] Proceedings of ACM Symposium on Computational Geometry, pp. 279-288, 1988.
19. Wolovich, W., A., Elliot, H., *A computational technique for inverse kinematics*, [in] Proceedings of 23rd IEEE Conference on Decision and Control, pp. 1359-1363, 1984.
20. Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., Tsang, E., *Prediction-Based Population Re-initialization for Evolutionary Dynamic Multi-objective Optimization*, [in] Lecture Notes in Computer Science, 4403, pp.832-846, 2007.