Available online at www.sciencedirect.com
SCIENCE DIRECT®
Robotics and Computer-Integrated Manufacturing 21 (2005) 67–79
Robotics and Computer-Integrated Manufacturing
www.elsevier.com/locate/rcim

Optimal robot task scheduling based on genetic algorithms

P.Th. Zacharia, N.A. Aspragathos*

Mechanical Engineering and Aeronautics Department, University of Patras, 26500 Patras, Greece

Received 10 April 2003; received in revised form 10 January 2004; accepted 25 April 2004

Abstract

Industrial robots should perform complex tasks in the minimum possible cycle time in order to obtain high productivity. The problem of determining the optimum route of a manipulator's end effector visiting a number of task points is similar but not identical to the well-known travelling salesman problem (TSP). Adapting TSP to Robotics, the measure to be optimized is the time instead of the distance. In addition, the travel time between any two points is significantly affected by the choice of the manipulator's configuration. Therefore, the multiple solutions of the inverse kinematics problem should be taken into consideration.

In this paper, a method is introduced to determine the optimum sequence of task points visited by the tip of the end effector of an articulated robot and it can be applied to any non-redundant manipulator. This method is based on genetic algorithms and an innovative encoding is introduced to take into account the multiple solutions of the inverse kinematic problem. The results show that the method can determine the optimum sequence of a considerable number of task points for robots up to six-degrees of freedom.

Keywords: Scheduling; Robots; Travelling salesman problem; Genetic algorithms

1. Introduction

Current manipulators should be capable of performing complex tasks in minimum time in order to increase the productivity. Thus, one of the objectives of the robotics research community is to develop algorithms that enable the manipulators to perform the desired tasks as quickly as possible, taking into account the limits imposed by their physical characteristics. The robot application considered in this work is composed of movements between points where the order of visit is not critical. Some examples of such applications are the insertion of electronic components, the laser cutting, the multiple drilling and the spot welding.

In many industrial operations, the order with which the manipulator will 'visit' a number of task points and return to the starting point is not predetermined. On the other hand, that order strongly affect the total cycle time required to visit the task points. Determining the optimum sequence means searching for the order that minimizes the execution time of the pre-assigned tasks.

The problem of determining the optimum sequence of manipulator's task points can be considered as an extension to the well-known travelling salesman problem (TSP) [1]. The TSP is one of the most widely discussed problems in combinatorial optimization. The salesman has to visit exactly once each one of a finite number of cities and return to the starting city. Given the distances between the cities, the objective of the optimization is to find the optimum total tour the salesman should follow. Adapting TSP to robotics, the measure to be minimized is the time instead of the distance. In other words, it is asked to determine the path which ensures that the robot pass through the given points following the order with the minimum possible cycle time. It should be stressed that the problem of finding the minimum cycle time takes into consideration the multiple solutions of the inverse kinematics problem. This implies that the choice of the manipulator's configuration significantly affects the global cycle time. So, this problem is more difficult and of higher complexity than the classic TSP problem.

*Corresponding author. Tel.: +30-2610-997-268; fax: +30-2610-997-744.

E-mail address: asprag@mech.upatras.gr (N.A. Aspragathos).

0736-5845/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2004.04.003

Several optimization techniques have been proposed to determine the minimum cycle time of a manipulator visiting a number of task points. The most prominent of these techniques are presented and discussed in the following.

Dubowsky and Blubaugh [2] used Little's algorithm to determine the minimum time to accomplish a manipulator task. The authors solved the TSP by distinguishing three cases. They solved the TSP where the task points may be visited in any order and tested their algorithm using a PUMA 260 manipulator, which has to 'visit' six points. In the second case, they solved the TSP, where some points may be visited in any sequence, whereas some others must be visited in an exact sequence. Lastly, they investigated tasks, which are not in the usual TSP form, including tasks where the manipulator has to change its tools during an operation. In all these TSP problems, the computation time required was quite reasonable, but there is no reference to the multiple solutions of the inverse kinematics problem.

Abdel-Malek and Li [3] developed a method for finding the optimum sequencing of the robotic task performance. The time is the performance measure and the minimum cycle time is determined taking into consideration the multiplicity of robot configurations. The algorithm is tested for three-degrees-of-freedom (3-DOF) robots (cartesian, cylindrical, spherical and articulated) having at maximum two different configurations for each point, which have to visit six points in the three-dimensional space. However, this algorithm was not extended for robots with more than two configurations and there is not any indication that can be extended by further elaboration.

The nearest neighbor (NN) algorithm is an algorithm that reduces the computational time and converges even for a large number of points. Edan et al. [4] applied the NN algorithm for fruit-harvesting robots in order to find the near-optimum-time path between the fruit locations. Shin and Mckay [5] have shown that the total time t_c to pass along the path between two points is proportional to the geodesic distance S. So, the cost function was defined as the distance along the geodesic in the inertia space, since calculating the geodesic distance in this space is equivalent to calculating time. The algorithm was implemented for a cylindrical robot with 4-DOF, which has to collect fruits from 20 trees in the three-dimensional space. The maximum number of fruits, for which the manipulator was tested, was 250 and the authors claimed that the computing time is acceptable for real-time applications. Furthermore, the suggested algorithm takes into account the kinematic and dynamic properties of the robot when determining the path of the robot through the task points. However, it is not mentioned whether the possible configurations are taken into account during the search for the optimum path. In conclusion, a fast algorithm is derived with a time complexity of only $O(N^2)$, where N is the number of the fruit points, but this method does not always result in the shortest tour. This means that the NN algorithm does not guarantee that the optimum solution will be found.

Simulated annealing (SA) is a technique derived from statistical mechanics and it is motivated by an analogy to the behavior of the physical annealing process. Dissanayake and Gal [6] used SA for determining a near-optimum sequence of travel for redundant manipulators. Because of the non-linearity of the optimization problem, the method of sequential quadratic programming is used to find robot's optimum configurations and the optimum location in the workcell for a given sequence of tasks. SA was applied for 3-DOF planar robots (3R, RPR and PRR), which have to pick up a tool, then travel through nine locations in the two-dimensional space and finally return the tool to its original position. The algorithm searched about 18 000 sequences for the nine work-site problem till convergence was obtained. However, it is estimated that for the 19 work-site problem, the algorithm should search about 45 000 sequences. So, a really powerful computer is required in order to obtain a solution within a reasonable time. Nevertheless, SA is a robust technique that performs well-solving complex problems and does not need much computational time when the number of the task points is less than 10.

Petiot et al. [7] used the elastic net method (ENM) in order to minimize the cycle time of robotic tasks. The method is well adapted to the problem of finding the optimum sequence of a manipulator's end effector moving from point to point. This is achieved by minimizing an energy function E using a modified gradient method. However, the algorithm may not converge due to wrong choice of the algorithmic parameters and this happened in 5% of the tests. The tests were run on a VAX 4500 computer. In the same paper, there is a comparison of the results between ENM and the Little's branch and bound algorithm proposed by Dubowski and Blubaugh [8]. As far as CPU time is concerned, the ENM is faster than Little's method. Nevertheless, the authors admitted that the ENM works well in the case of 2- or 3-DOF robots but it is very difficult to be used in the case of robots with more than three degrees of freedom because of the increased computer time cost.

The algorithms mentioned in the previous paragraphs work well for 2- or 3-DOF manipulators, but they have difficulties and increase the time cost or cannot be generalized in the case of manipulators with more degrees of freedom. Some of these algorithms take into consideration the multiple solutions of the inverse kinematics problem. In all the examples presented in the discussed papers, each point in the two-dimensional

space can be reached with two configurations. In addition, these algorithms do not require too much time to find the optimum solution for a number of points up to 10; however, the time cost is significantly increased for more than 10 points.

During the last two decades several approaches have been proposed for the solution of the TSP and in some cases were modified to determine the minimum cycle time of a manipulator. The approach introduced in this paper is based on the very promising and rapidly developing genetic algorithms (GAs) in order to find the optimum sequence of a manipulator's end effector moving between the task points. Despite the fact that several researchers [9–11] have dealt with the TSP using GAs, they have not taken into consideration robots and their multiple configurations.

The main advantage of the GAs over the gradient-optimization methods is that they do not require the derivative of the objective function. So, they are able to find the near global optimum of non-continuous and procedural functions. In addition, GAs can be easily understood with very few mathematics.

Rekleitis and Aspragathos [12] found the optimum sequence using GAs, where the objective function was the travel time. The genotype of the GA was an array of integer numbers that demonstrates a potential solution to the problem. The proposed method computes all the combinations of the possible configurations resulting from the inverse kinematics problem with which the robot's end effector can reach the N points and finally selects the one that gives the minimum travel time. In their approach, they take into consideration the time required to move the joints from the configuration with which the robot reaches a task point to the configuration with which it leaves this point. The GA is implemented for the case of a PUMA-like robot with four different configurations and results were found for seven and 14 points in the three-dimensional space. The results show that the GA is a reliable procedure, despite the fact that it is relatively slow. However, this is not a serious drawback, since the robot scheduling is an off-line procedure.

This paper introduces a method based on GAs for the determination of the optimal sequence of a non-redundant manipulator's end effector, i.e. the sequence that guarantees the minimum cycle time, taking into consideration its multiple configurations. In our approach, each chromosome of the population is formed in such a way that the first part represents the sequence with which the manipulator reaches the N task points and the second part represents the manipulator's configurations derived by the solution of the inverse kinematics in each task point. The integer alphabet is used for the first part, whereas for the second part the binary and the integer alphabet is used. The proposed algorithm is general and can be applied to any non-redundant manipulator without regard to its complexity, taking into account the multiple solutions of the inverse kinematics problem.

The remainder of the paper is organized as follows. Section 2 defines the objective function that has to be minimized. Section 3 briefly describes the basic structure of GAs. In this paragraph, the proposed encoding mechanism is presented in detail. Section 4 presents the results and analyses the efficiency of the proposed GA and the paper concludes with Section 5.

2. Formulation of the objective function

The relation between the vector $\mathbf{s}$, denoting the position and orientation of the end effector, and the joint displacement vector $\boldsymbol{q}$ is generally non-linear. This kinematic relation of the manipulator is given by

$$\boldsymbol{s} = \boldsymbol{f}(\boldsymbol{q}). \tag{1}$$

When the joint displacements represented by the vector $\boldsymbol{q}$ are given, the corresponding $\mathbf{s}$ is determined uniquely and the calculation is rather simple. When a task is assigned to the manipulator, its end-effector position and orientation $\mathbf{s}$ or a trajectory $\boldsymbol{s}(t)$ is given, then the solution of Eq. (1) can be written formally as

$$\boldsymbol{q} = \boldsymbol{f}^{-1}(\boldsymbol{s}). \tag{2}$$

However, a solution $\boldsymbol{q}$ does not necessarily exist and even when it does exist, it is not usually unique. The problem of obtaining $\boldsymbol{q}$ corresponding to a given $\boldsymbol{s}$ is called the inverse kinematics problem.

The structural characteristics of the robot (the anatomy, the type of the joints, the length of its links, the joint velocity, etc.) affect significantly the time spent on travelling between any two particular task points. Inverse kinematics can be used to compute the travel time between two points.

This paper deals with the following problem: Let a manipulator, which has to visit N points (exactly once each one) and return to the initial point. It is asked to determine the tour the manipulator has to travel, so that the total cycle time is minimized.

At this point, it should be mentioned that in the case of non-redundant robots, one point of the operational space can be reached with a finite number of configurations. This means that in the case of multiple configurations, the optimum sequence is certainly affected by the configuration choice. So, given the joint displacements for each configuration, it is easy to compute the time needed for travelling between any two points. The total time needed for the completion of a tour constitutes the objective function to be optimized. In the following paragraphs, the formulation of this function is described in detail.

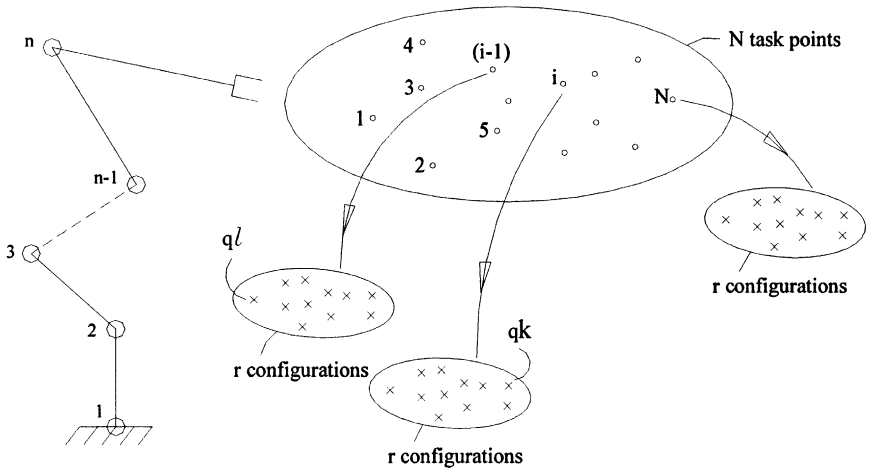


Fig. 1. A schematic of a n-DOF manipulator, the N task points and the r configurations corresponding to every task point.

Let an n-DOF manipulator operating in the m-dimensional space ($n \leqslant m$) that has to visit N task points, the locations of which are known and r is the number of solutions of the inverse kinematics problem. Fig. 1 shows an n-DOF manipulator, which has to visit N task points in the m-dimensional space. A set of r configurations (expressing the r solutions of the inverse kinematics problem) corresponds at each one of the N task points. The objective is to find the tour for visiting all the task points (once each one), so that the minimum travel time is determined, taking into account the multiplicity of the robot configurations corresponding to every task point.

First of all, the joint coordinates ($\boldsymbol{q} \in R^n$) are computed solving the inverse kinematics problem for each point of the m-dimensional space. Therefore, the time t_i spent by the manipulator to travel from the task point $(i-1)$ using the $\boldsymbol{q}^{\ell}$ set of the joint coordinates, which determines the ℓ-configuration, to the task point i using the $\boldsymbol{q}^{k}$ set of the joint coordinates, which determines the k-configuration, can be written as

$$t_i = \max\left(\frac{|\boldsymbol{q}_{ji}^{k} - \boldsymbol{q}_{j(i-1)}^{\ell}|}{\dot{\boldsymbol{q}}_j}\right), \quad \begin{matrix} j = 1, 2, \ldots, n \\ k, \ell = 1, 2, \ldots, r \end{matrix} \tag{3}$$

where $\boldsymbol{q}_{ji}$ is the jth joint displacement for the ith end-effector location and $\dot{\boldsymbol{q}}_j$ is the average velocity of the joint j. Eq. (3) denotes the fact that the travel time between two positions is determined by the slowest manipulator's joint. It is also clear that for the case of multiple solutions, the choice of the manipulator's configuration affects significantly the travel time. So, the total travel time t_c required to visit all the task points is given by

$$t_c = \sum_{i=2}^{N} t_i \tag{4}$$

and the optimization problem can be written as

$$t_{opt} = \min \sum_{i=2}^{N} t_i. \tag{5}$$

The optimization variables of this function are the robot configurations corresponding to each task point. The optimum value of the function, expressed by Eq. (5), represents the minimum travel time between N points. The search space of this function is discrete, because the number of all the task points the robot has to visit is finite. It is clear that the function is non-continuous, non-linear and procedural.

At this point, we should notice that the problem is defined in the configuration space in this method, so the complexity of the problem increases, since it depends on the number of the task points and the number of the possible manipulator's configurations. Tsai and Morgan [13] proved that the inverse kinematics problem of a six-revolute joint manipulator has at most 2^8 ($=256$) solutions. In the special case where the last three axes intersect, the possible configurations are proved to be 2^4 ($=16$). Craig [14] based on this paper marked that the maximum number of solutions is related to how many of the link length parameters are zero. So, in this case the maximum number of solutions is expressed as 2^d, where $d = 1, 2, 3, 4$. The number of all the possible tours is $(N-1)!/2$ where N is the number of the points. The number of all the possible configurations with which the manipulator can reach the N points is $(2^d)^N$ for each tour, where 2^d are the solutions of the inverse kinematics problem. Consequently, the range R_s of the search space is

$$R_s(N, d) = \frac{(N-1)!}{2}(2^d)^N. \tag{6}$$

For example, for a 2-DOF manipulator operating in the two-dimensional space that has to visit 10 points with two different configurations corresponding to each

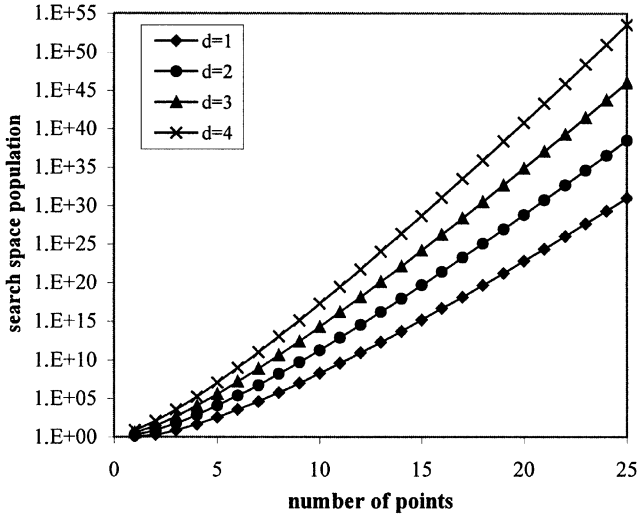


Fig. 2. The search space population versus the number of points for 2, 4, 8, 16 configurations.

point, the population of the search space is $R_s(10,1) = 1.85 \times 10^8$. For the case of a 6-DOF manipulator operating in the three-dimensional space that has to visit 10 points with eight different configurations corresponding to each point, the population of the search space is $R_s(10,3) = 1.94 \times 10^{14}$. It is clear that the number of the possible solutions depends at a great extent on the number of the possible configurations. Fig. 2 represents the population of the search space versus the number of the task points, for the cases where the manipulator has to visit a point in the space with 2, 4, 8 and 16 possible configurations.

3. Optimization based on GAs

GAs are adaptive search techniques based on the principles and mechanisms of natural selection and the 'survival of the fittest'. GAs grew out of Holland's study of adaption in artificial and natural systems. The GA works in parallel a certain number of chromosomes and each of them is made of a finite string of symbols (genes) and represents a possible solution for a given objective function. The set of chromosomes, which is changed at each iteration (generation), is called population, and the members of a population are called individuals.

The GA, like any other optimization algorithm, begins with the determination of the fitness function and the control parameters. It stops, like any other optimization algorithm, checking for convergence. The components of the GA are outlined below.

3.1. The representation mechanism

The representation mechanism is the main innovation of the proposed approach, so it is described in detail. The first step in applying the GA is the choice of an appropriate representation to encode the possible solutions of the current optimization problem. The solution is represented as a string over a specific alphabet.

The bit string representation of solutions has dominated GA research, since the binary alphabet offers the maximum number of schemata per bit of information of any encoding [15]. Holland [16] gave a theoretical justification for using binary encoding. He compared the number of schemata available in a binary coding to the number of schemata available in a non-binary coding, where the two encodings had the same information-carrying capacity. Holland's schema-counting argument seems to imply that GAs exhibit worse performance on multiple-character encodings than on binary encodings. However, several tests showed that non-binary encodings perform better than the binary encodings [17].

Consequently, the performance depends very much on the problem, because binary encodings are unnatural and unwieldy for many problems and are prone to rather arbitrary orderings. Nevertheless, there are no rigorous guidelines for predicting which encoding will work best.

As far as the TSP is concerned, the binary representation of tours is not well suited, since the binary code of the cities will not provide any advantage. On the contrary, the binary representation would require special repair algorithms, since a change of a single bit may result in an illegal tour [18].

In the proposed GA, each chromosome consists of two parts. The first part represents the order of the task points, whereas the second part includes the robot configuration corresponding to each task point.

For the case of a 2-DOF manipulator that has to visit N points in the two-dimensional space with two different configurations corresponding to each point, each chromosome consists of $2N$ genes. The first part of the string composed of N symbols, represents the sequence with which the manipulator reaches the N task points. For this part, the integer alphabet is used. The second part of the string composed of N symbols, represents the manipulator's configuration. For the second part, the binary alphabet is used. So, given the lengths of the links L_1 and L_2 and the position coordinates x, y of the N task points, the solution of the inverse kinematics problem for the joint variables $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are [14]:

$$\boldsymbol{q}_1 = \tan^{-1}(y/x) \pm \cos^{-1}\left(\frac{x^2+y^2+L_1^2-L_2^2}{2L_1\sqrt{x^2+y^2}}\right) \text{ and}$$

$$\boldsymbol{q}_2 = \cos^{-1}\left(\frac{x^2+y^2-L_1^2-L_2^2}{2L_1L_2}\right). \quad (7)$$

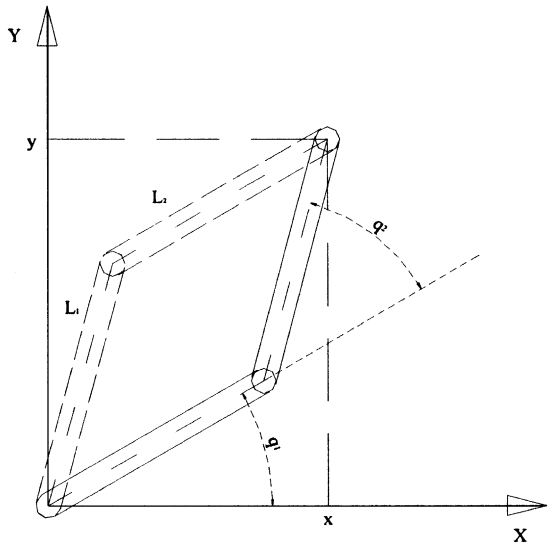
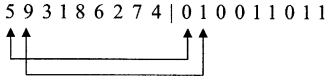


Fig. 3. Two different configurations of a 2-DOF manipulator operating in the two-dimensional space.

The $\pm$ symbol implies that there are two different solutions for the joint variable $\mathbf{q}_1$ resulting in two different configurations. So, the first configuration (continuous line in Fig. 3) has the solution ($\boldsymbol{q}_{1-}$, $\boldsymbol{q}_2$), whereas the second configuration (dashed line in Fig. 3) has the solution ($\boldsymbol{q}_{1+}$, $\boldsymbol{q}_2$). In other words, the value of the binary digit corresponding to the task point in the two-dimensional space determines one out of the two configurations of the manipulator. In particular, the configuration ($\boldsymbol{q}_{1-}$, $\boldsymbol{q}_2$) corresponds to the binary digit 0, whereas the configuration ($\boldsymbol{q}_{1+}$, $\boldsymbol{q}_2$) corresponds to the binary digit 1.

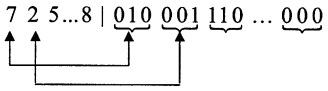
Let a 2-DOF manipulator that has to visit 9 points in the two-dimensional space, then the chromosome can be formed as



which means that the manipulator reaches task point 5 with the configuration corresponding to the binary digit 0, then it reaches point 9 with the configuration corresponding to the binary digit 1, etc.

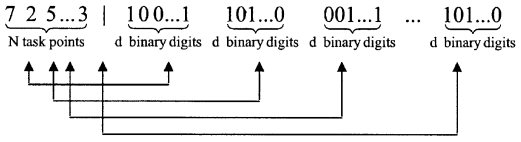
Let a 6-DOF manipulator that has to visit N task points in the three-dimensional space with eight different configurations corresponding to each task point. The first part of the string composed of N symbols, represents the sequence with which the manipulator reaches the N points. For this part, the integer alphabet is used. The second part of the string composed N bytes of 3 bits, represents the manipulator's configuration.

In other words, each byte of (000, 001, 010,…, 111) corresponding to a task point in the three-dimensional space, determines one out of eight configurations of the manipulator, in a similar way with the one presented for the case of the 2-DOF manipulator. Let a manipulator that has to visit 9 points in the three-dimensional space, the chromosome can be formed as



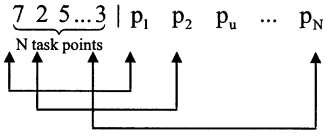
which means that the manipulator reaches point 7 with the configuration corresponding to the byte 010, then it reaches point 2 with the configuration corresponding to the byte 001, etc.

In general, for a manipulator that has to visit N points in the m-dimensional space, reaching each task point with 2^d different configurations, the chromosome is formed as follows:



where the arrows show the correspondence of one configuration to the respective task point.

For the second part, an integer encoding is used for comparison. For this case, the chromosome is formed as follows:



where p_u is an integer number varying between 1 and 2^d and corresponds to a specific configuration.

At this point, it should be made discrimination between the integer encoding used to represent the TSP and the integer encoding used for the robot's configurations. The integers corresponding to the task points that the robot has to 'visit' cannot be repeated, since the robot has to reach each task point exactly once. On the other hand, the robot can reach the task points with any configuration, so the integers corresponding to the robot's configurations can be repeated. This implies that the genetic operators (crossover and mutation) applied to the two parts of the chromosome cannot be the same.

Despite the fact that the binary encoding seems to be more complex than the integer encoding, it performs much better. This is practically confirmed by the results from the experiments shown in Tables 3 and 4. In addition, the code for the binary encoding is simpler, because it is not necessary to correspond a binary number with a robot's configuration (q_1, q_2,…), as it is explained above in this paragraph.

3.2. Initial population

Usually, the initial population is randomly selected in order to uniformly distribute the selected chromosomes (solutions over the search space). In some cases, solutions obtained from another optimization algorithm are used to seed the initial population [19]. Although this bears the risk of misguiding the optimization process toward local optima, it has been proved that the seeding approach is very powerful in some cases. In our approach, the initial population is seeded using case retrieval in order to speed up the GA in finding a solution [20].

3.3. The evaluation mechanism

The evaluation mechanism uses the objective function of the current problem, which has to be minimized. This function is transformed to the fitness function, which is evaluated for all the chromosomes of the population. The value of the fitness function for one chromosome is a reflection of how well this chromosome is adapted to the environment, i.e. the ability of the chromosome to survive and be reproduced in the next generation.

In the problem discussed here, the objective function deals with the time needed by the manipulator to visit N task points and return to the starting point and the goal is the minimization of this time. The procedure to obtain the objective function is described in detail in Section 2. The fitness function is the inverse of the objective function

$$\text{fitness} = \frac{1}{t_c}, \tag{8}$$

where t_c is given by Eq. (4). Since $t_c \neq 0$, it is not necessary to take any precaution to avoid infinite values.

3.4. Reproduction

Reproduction is an operator that follows the process of natural selection and the 'survival of the fittest'. It is a simple copy of an individual from one generation to the next one without any modification. Reproduction has the purpose of preserving the good traits, carried by the good individuals of the population, and spreading them over the population at a higher rate. Consequently, reproduction does not produce new individuals.

In the proposed GA, the chromosomes are copied from the previous generation to the next one according to the normalized (and not the absolute) values of the fitness function. The proportional selection is based on the roulette wheel strategy, where the chromosomes that will be copied are selected with rates proportional to their fitness. This means that the probability is higher for a chromosome with high fitness to be selected for reproduction than another with lower fitness.

3.5. Crossover

Crossover is a recombination operator and follows the reproduction. The role of this operator is to join together parts of several individuals in order to produce new ones for the next generation. The individuals are randomly selected according to a predefined probability (crossover rate). The crossover operator used for the first part of the chromosome, which is consisted of decimal digits, is the order crossover (OX) [21] and for the second part of the chromosome, the one-point crossover is used [18].

3.6. Mutation

Mutation is applied performing a random modification on some individuals with a small-predefined probability (mutation rate). The mutation operator used for the first part of the chromosome consisted of decimal digits is the inversion [21]. Inversion is simply applied to a chromosome and guarantees that the resulting offspring represents a 'legal' tour. For the second part of the chromosome, the mutation operator is applied changing a random gene of digital value '0' to '1' and vice versa for the binary encoding, whereas for the integer encoding a random gene changes to another value between 1 and 2^d.

3.7. Control parameters

A number of control parameters that affect the GA process and consequently the convergence rate and the final result has to be defined. The most important control parameters are the following:

Population size: The population size determines the number of the chromosomes and therefore how much genetic material is available during the genetic search. It has to be mentioned that a small population size covers a small area of the search space, which means that it may not be a representative sample of the solutions. So, a small population size decreases the possibility of finding a global optimum. On the other hand, a large population size significantly increases the CPU time. The population size depends on the nature and the complexity of the current problem. In this work, the proposed algorithm was tested for various population sizes. Finally, the selected population size is equal to 200 for the case of a 3-DOF manipulator and 500 for a 6-DOF manipulator.

Crossover rate: The crossover rate determines the frequency with which the crossover operator is applied to the chromosomes of the population, so that a new population is generated. The higher the crossover rate is, more individuals are introduced in the new population. The crossover rate is usually in the range between 0.6

and 1.0 and in our case, it is selected equal to 0.8 after a considerable number of trials.

Mutation rate: The mutation rate determines the probability that a gene's value in a chromosome would be changed. Mutation introduces new areas of the unexplored search space. However, the mutation rate should not be too high, because it increases the randomness in the search. The mutation rate is usually less than 0.4 and in our case, it was selected equal to 0.1 after too many trial runnings.

Elitism: Elitism is the selection strategy that guarantees the survival of the best chromosome of the population to the next generation. This is achieved by comparing the best chromosome of the current generation with the best one of the previous generation and preserving the best of the two chromosomes.

3.8. Termination conditions

There is not mathematical proof of convergence or any guarantee that the GA will find the global optimum. In addition, it is not clear which is the best way to terminate the algorithm. In many cases, a maximum number of iterations (generations) is defined in advance. However, the predetermination of the maximum number of generations implies that the duration of the genetic search is fixed, regardless of the search success. Moreover, it is difficult to determine beforehand the number of generations needed to find near-optimum solutions. Thus, an assessment of the quality level of the GA should be made on-line.

In our approach, the condition to be satisfied so that the evolution is aborted is the iteration of the same solution for a predefined number of generations. So, the algorithm terminates by defining in advance the number of iterations (generations) for which the same chromosome constantly appears as the optimum one. It should be mentioned that the best solution appears for several iterations before a fortuitous crossover or mutation produces a better solution. For this reason, the maximum number of iterations should be large enough; otherwise, a misleading result may arise.

4. Discussion of results

In this section, the method for optimal robot task scheduling tested in four sets of simulated experiments. In paragraph 4.1, the two encodings, namely binary and integer, for the second part of the chromosome are tested in order to compare the best cycle times achieved in each case. In paragraph 4.2, the influence of the control parameters on the results of the proposed GA for the case of a 3-DOF and a 6-DOF manipulator is studied. In paragraph 4.3, the results of the proposed approach are compared with other methods presented in the introduction. In all experiments presented in the following paragraphs, the term 'near-optimum' is adopted for the best solution found, since it cannot be proved that the solution found was the optimum solution.

4.1. The encoding of the proposed GA

In this paragraph, the problem is solved using the two encodings for the second part of the chromosome.

For both encodings, the integer alphabet is used for the first part of the chromosome. For the second part of the chromosome, the binary alphabet is used for the first encoding, whereas for the second encoding the integer alphabet is used.

The algorithm was tested using these two encodings for a 3- and a 6-DOF manipulator for the same points and the same control parameters. The seeding percentages for each case are selected after several runnings of the GA. Tables 1 and 2 show the near-optimum solutions and the generation at which each one appears for the first time for several different percentages. The results shown in Tables 1 and 2 correspond to the case of a 3-DOF and a 6-DOF robot that has to reach 10 task points in the three-dimensional space. It is clear from the results that the seeding percentage influences the near-optimum solution the algorithm finds and it is different for each case. So, the best 'near-optimum' solution for the case of a 3-DOF robot is found for a seeding percentage of 70%, whereas for the case of a 6-DOF robot the seeding percentage is 45%.

The results shown in Tables 3 and 4 confirm that the binary encoding representing the configurations of the manipulator is better than the integer encoding for almost all the cases comparing the cycle time and the

Table 1
Near-optimum solutions versus seeding percentages for a 3-DOF robot

Seeding percentage	Near-optimum solution (s)	Generation of the near-optimum solution
5	3.14	72
10	3.24	2
15	3.09	196
20	3.01	96
25	2.92	358
30	2.92	38
35	3.06	31
40	3.27	11
45	3.01	155
50	3.20	44
60	3.14	160
70	**2.92**	**35**
80	3.08	184
90	3.25	52
100	3.24	12

Table 2
Near-optimum solutions versus seeding percentages for 6-DOF robot

Seeding percentage	Near-optimum solution (s)	Generation of the near-optimum solution
5	5.07	166
10	6.26	2
15	5.87	7
20	6.26	2
25	6.26	2
30	6.26	2
35	6.10	3
40	6.26	2
45	**4.82**	**3**
50	4.66	125
60	4.66	241
70	6.26	2
80	6.05	219
90	6.10	3
100	6.10	3

Table 3
Cycle times using binary and integer encoding 3-DOF manipulator

Number of points	Binary encoding		Integer encoding	
	Cycle time (s)	Generation	Cycle time (s)	Generation
5	2.31	Third	2.35	18th
10	2.92	35th	4.86	48th
15	3.10	First	8.65	212th
20	3.57	14th	9.57	140th
25	4.02	87th	10.62	123th

Table 4
Cycle times using binary and integer encoding 6-DOF manipulator

Number of points	Binary encoding		Integer encoding	
	Cycle time (s)	Generation	Cycle time (s)	Generation
5	1.72	First	1.72	First
10	4.82	Third	6.36	Third
15	15.11	95th	14.58	265th
20	19.81	91th	19.85	445th
25	24.37	106th	26.02	312th

generation at which the GA converges. So, the binary alphabet is used for the second part of the chromosome for all the tests implemented in paragraphs 4.2 and 4.3.

4.2. The influence of the control parameters

The proposed GA is applied to a 3-DOF manipulator that has to visit 10 points in the three-dimensional space. Various combinations of the control parameters are tested and the maximum number of generations of the same chromosome is set to 200. Table 5 shows the values of the control parameters, the resulting near-optimum solutions and the generation in which the near-optimum solution appears for the first time for a selected algorithm runnings out of the considerable number of experiments carried out in this work.

The combination of the control parameters which gives the 'best' near-optimum solution is: population size = 450, crossover rate = 0.9 and mutation rate = 0.25 and the minimum cycle time is 2.92 s. This near-optimum solution appears for the first time in the 35th generation. The worst cycle time, which is achieved for the control parameters of the first row, is 3.32 s, which means that it is

$$\frac{|3.32-2.92|}{2.92}100\% = 13.69\%$$

greater than the best one. Fig. 4 shows the best, the average and the worst values of the total cycle time during the convergence of the GA, with the above values of the control parameters.

In the second part of the simulated experiments, the proposed method is tested for a 6-DOF manipulator that has to visit 10 points in the three-dimensional space. A variety of control parameters are tested, whereas the algorithm terminates after 200 iterations (generations) of the same chromosome. Selected results are shown in Table 6.

In this case, the combination of the control parameters, which gives the 'best' near-optimum solution is: population size = 400, crossover rate = 0.8 and mutation rate = 0.1 and the minimum cycle time is 4.82 s. This near-optimum solution appears for the first time in the third generation. The worst cycle time is 8.32 s, which is

$$\frac{|8.32-4.82|}{4.82}100\% = 72.61\%$$

greater than the best one. Fig. 5 shows the best, the average and the worst values of the total cycle time during the convergence of the GA.

Considering the results of these trials, simulated experiments are assigned to the control parameters: crossover rate = 0.9, mutation rate = 0.25 and population size = 450 for a 3-DOF and crossover rate = 0.8, mutation rate = 0.1 and population size = 400 for a 6-DOF manipulator, respectively.

Fig. 6 depicts the optimum sequence of 3-DOF manipulator's route visiting 10 points located on the x–y plane with the above values of the control parameters, in order to show a simple example of the optimum sequence found by the introduced method.

4.3. Computational time results

Computational time results of previous approaches are very limited in the relevant literature. So, it is difficult to present an extended comparison. In addition, the computers used in previous works are different from the computers used in this work. Despite these

Table 5
Near-optimum solutions versus control parameters
3-DOF

Population size	Crossover rate	Mutation rate	Near-optimum solution (s)	Generation of the near-optimum solution
150	0.8	0.10	3.32	Fourth
150	0.9	0.25	3.32	Fourth
250	0.8	0.10	3.01	Third
250	0.9	0.25	3.16	59th
350	0.8	0.10	3.13	13th
350	0.9	0.25	3.29	111th
450	0.8	0.10	3.01	53th
450	**0.9**	**0.25**	**2.92**	**35th**

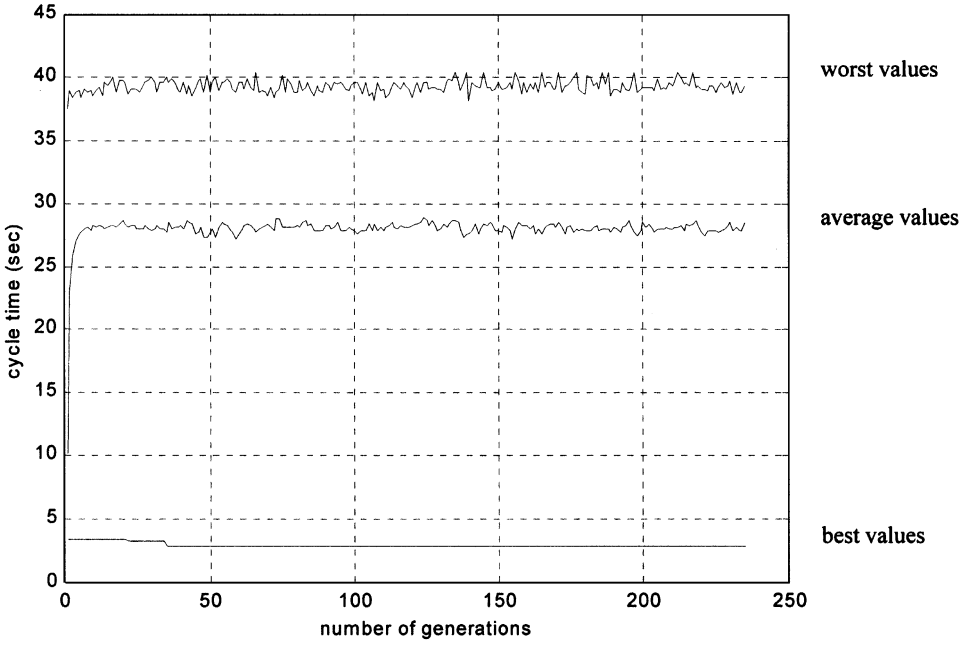


Fig. 4. Convergence of the proposed GA for the case of a 3-DOF manipulator.

Table 6
Near-optimum solutions versus control parameters
6-DOF

Population size	Crossover rate	Mutation rate	Near-optimum solution (s)	Generation of the near-optimum solution
200	0.8	0.10	5.48	130th
200	0.9	0.25	8.32	First
300	0.8	0.10	4.82	34th
300	0.9	0.25	7.36	224th
400	**0.8**	**0.10**	**4.82**	**Third**
400	0.9	0.25	7.59	33th
500	0.8	0.10	6.67	Second
500	0.9	0.25	8.32	First

difficulties some computational results can give an indication of the advanced performance of the proposed approach.

In [7], the elastic net method is applied for a 2-DOF manipulator (one prismatic joint and one rotational joint), which has to 'visit' 10 points in the two-dimensional space. This manipulator can reach a point in the space with two different configurations resulting from the solution of the inverse kinematics problem. The CPU time was 20 s and the algorithm run on a VAX 4500 computer, whereas the estimated CPU time using Little's method was 1280 s.

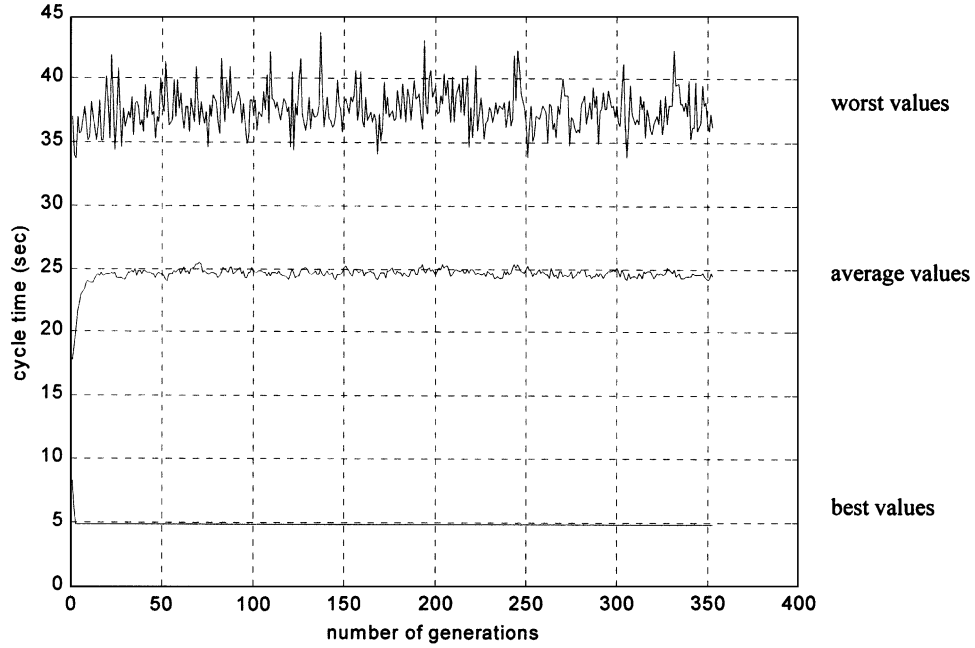


Fig. 5. Convergence of the proposed GA for the case of a 6-DOF manipulator.

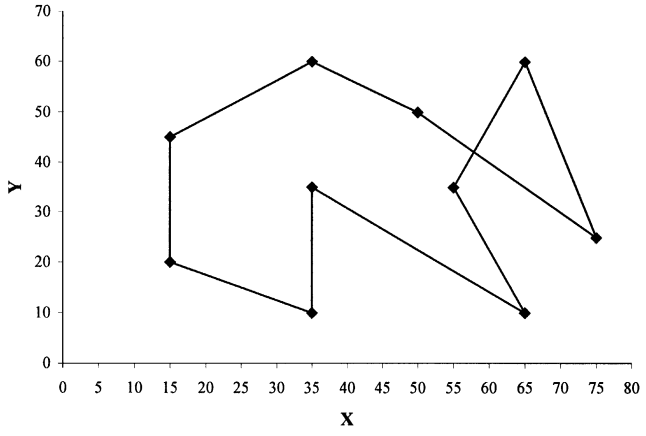


Fig. 6. Optimum sequence for 10 points on the x–y plane for a 3-DOF manipulator.

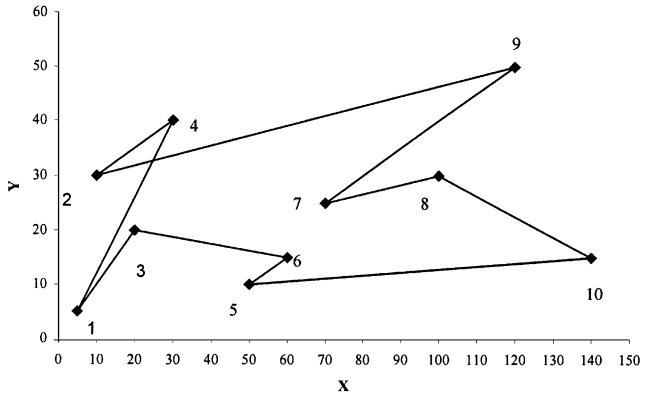


Fig. 7. Optimum route for 10 points on the x–y plane for a 2-DOF manipulator.

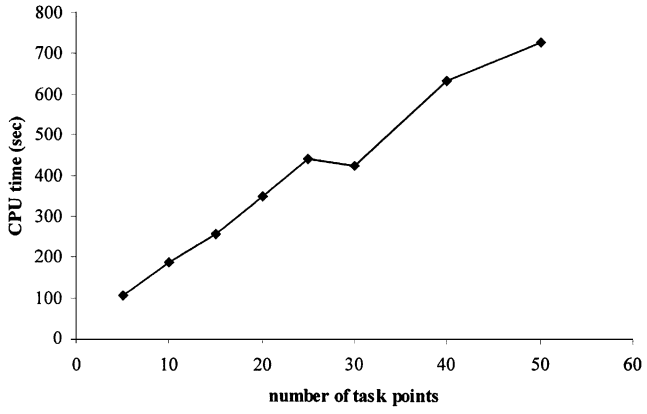


Fig. 8. CPU time versus the number of task points in the 3-D space for a 3-DOF manipulator.

Using the proposed approach, the optimum sequence of the end-effector's route visiting ten points in the two-dimensional space is shown in Fig. 7. The CPU time is 3.91 s and the algorithm run on a Pentium 4 PC at 1.8 GHz under Matlab platform.

An interesting indication of the computational time performance is the variation of the calculation time versus the number of task points. Figs. 8 and 9 show this variation which is approximately linear with the computational time results. The CPU time does not increase at a great rate versus the number of points. At this point, it should be stressed that the optimal robot-scheduling problem is much more complicated than the classical TSP problem, since the multiple configurations of the robot are also taken into consideration. The range

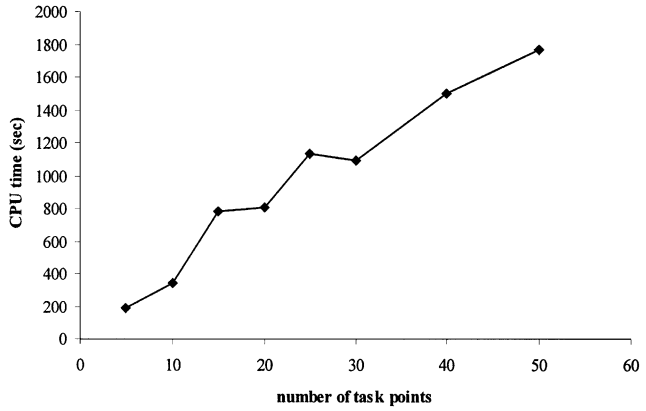


Fig. 9. CPU time versus the number of task points in the 3-D space for a 6-DOF manipulator.

of the search space is given by Eq. (6) and for the case of a 6-DOF manipulator that has to visit 10 task points with eight configurations, the possible solutions are $R_s(10,3) = 1.94 \times 10^{14}$, whereas for 20 task points the possible solutions explode to $R_s(20,3) = 7.01 \times 10^{34}$! So, despite the fact that the search space increases exponentially with the increase of the task points, the CPU time increases almost linearly.

Similar results are presented by Ahn et al. [22] for a GA solving the simple path routing problem, where the CPU time increases linearly with the number of points. Using Little's method and ENM [7], the computational time increases significantly (almost exponentially) with the number of task points, despite the fact that the multiple configurations are not taken into consideration.

5. Conclusions

In this paper, a method based on GAs is introduced to determine the optimum sequence of a manipulator's end effector. This method is based on GAs and the main innovation is made on the encoding of the GA in order to take into consideration the multiple solutions of the inverse kinematics problem for the computation of the total cycle time. In our approach, the chromosome consists of two parts. The first part of the string represents the sequence with which the manipulator reaches the N task points and the second part represents the manipulator's configuration corresponding to each task point. For the first part the integer alphabet is used, whereas for the second part the binary alphabet is used. The pattern with which the multiple solutions of the inverse kinematics problem are taken into account is general and can be easily applied to any non-redundant manipulator.

Concerning the structure of the GA, the choice of the control parameters has a great influence on the solution of the problem that is optimized. Some combinations of the control parameters are tried and the combination that gives the 'best' near-optimum solution is finally selected.

In conclusion, our proposed approach is found to be an effective and efficient method for determining the near-optimum sequence of a manipulator's end effector route visiting a number of task points taken into account the multiple solutions of the inverse kinematics problem. The proposed method proved to be rather fast in finding an optimum or a near-optimum solution within an affordable time. Another advantage is that the multiple configurations of any non-redundant manipulator are easily embodied in the encoding of the GA. Unlike some of the methods referred in the introduction, the proposed approach can be used to find the near-optimum sequence for a 6-DOF manipulator operating in the three-dimensional space within a reasonable CPU time.

At this point, it should be stressed that the problem is much more complex than the classical TSP problem, because the configurations of the robot are also taken into consideration.

Considering the future research work, the proposed algorithm can be extended so that it can take into account the obstacle avoidance. In addition, the encoding of the proposed GA could be applied in similar problems, where there is a finite number of ways to reach a point or to move from point to point.

References

[1] Lawer E, Lenstra J, Rinnooy Kan A, Shmoys D. The travelling salesman problem. Chichester, UK: Wiley; 1985.
[2] Dubowsky S, Blubaugh T. Planning time-optimal robotic manipulator motions and work places for point-to-point tasks. IEEE Conference on Decision and Control. Ft. Lauderdale, FL, 1985.
[3] Abdel-Malek L, Li Z. The application of inverse kinematics in the optimum sequencing of robot tasks. Int J Prod Res 1990;28(1): 75–90.
[4] Edan Y, Flash T, Peiper U, Schmulevich I, Sarig Y. Near-minimum-time task planning for fruit-picking Robots. IEEE Trans Robotic Autom 1991;7(1).
[5] Shin K, Mckay N. Selection of near minimum time geometric paths for robotic manipulators. IEEE Trans Automat Control 1986;31(6):501–11.
[6] Dissanayake M, Gal J. Workstation planning for redundant manipulators. Int J Prod Res 1994;32(5):1105–18.
[7] Petiot J, Chedmail P, Haschoët J-Y. Contribution to the scheduling of trajectories in robotics. Robotic Comput Integr Manuf 1998;14:237–51.
[8] Dubowski S, Blubaugh T. Planning time-optimal robotic manipulator motions and work places for point-to-point tasks. IEEE Trans Robotic Autom 1989;5(3):377–81.
[9] Hwang H-S. An improved model for vehicle routing problem with time-constraint based on genetic algorithm. Comput Ind Eng 2002;42:361–9.
[10] Qu L, Sun R. A synergetic approach to genetic algorithms for solving traveling salesman problem. Inform Sci 1999;117:267–83.

[11] Chatterjie S, Carrera C, Lynch L. Eur J Oper Res 1996;93:490–510.
[12] Rekleitis G, Aspragathos N. Optimization of the cycle time in a robotic workcell using genetic algorithms. NMCR 2001, Cardiff.
[13] Tsai L, Morgan A. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. Paper 84-DET-20. ASME Mechanics Conference. Boston, 7–10 October, 1984.
[14] Craig J. Introduction to robotics, 2nd ed. Reading, MA: Addison-Wesley; 1989. p. 116–120, 126–7.
[15] Goldberg D. Genetic algorithms in search. optimation and machine learning. Reading, MA: Addison-Wesley; 1989 p. 80-2.
[16] Holland JH. Adaption in natural and artificial systems. Michigan: University of Michigan Press (Second edition: Cambridge, MA: MIT Press; 1992).
[17] Mitchell M. An introduction to genetic algorithms, 2nd ed. Cambridge, MA: MIT Press; 1996. p. 156–8.
[18] Michalewitz Z. Genetic algorithms + data structures = evolution programs, 3rd ed. Berlin: Springer; 1996. p. 21, 219–20.
[19] Thomsen R, Fogel G, Krink T. A clustal alignment improver using evolutionary algorithms. Proceedings of the 2002 Congress on Evolutionary Computation—CEC'02. 2002. p. 309–14.
[20] Oman S, Cunningham P. Using case retrieval to seed genetic algorithms. Int J Comput Intell Appl 2001;1(1):71–82.
[21] Davis L. Applying adaptive algorithms to epistatic domains. Proceedings of the International Joint Conference on Artificial Intelligence. 1985. p. 162–4.
[22] Ahn CW, Ramakrishna RS. A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Trans Evol Comput 2002;6(6):556–79.