Time-optimal task scheduling for two robotic manipulators operating in a three-dimensional environment

E K Xidias, P Th Zacharia*, and **N A Aspragathos**
Department of Mechanical and Aeronautical Engineering, University of Patras, Rion, Greece

The manuscript was received on 14 November 2009 and was accepted after revision for publication on 23 June 2010.

DOI: 10.1243/09596518JSCE949

Abstract: The present paper introduces a method for determining the optimal task scheduling for a two-robot work cell. This problem is reminiscent of the classic Travelling Salesman Problem (TSP), but the measure to be optimized is the time instead of the distance. In addition, this is a much more complex problem, since it involves two robots (salesmen), which have to visit different task-points (cities) considering the multiplicity of the robots' inverse kinematics. The optimization problem addressed in this work concerns the determination of the (near-)optimum sequence of task-points that should be visited by each one of the two robots while ensuring minimum total cycle time and collision avoidance among their links. The proposed approach is based on genetic algorithms and a special encoding is used to incorporate the division of the task-points for both robots and the multiple solutions of the inverse kinematics. The method was tested in different scenarios and the experimental results demonstrated the efficiency and effectiveness of the proposed approach.

Keywords: multi-robot system, scheduling, optimization, robotic manipulator, genetic algorithms

1 INTRODUCTION

Industrial robots have made a significant contribution towards the automation of manufacturing processes. The efficient use of robots shows productivity increase, production cost reduction, and product quality improvement. However, only one robot in a common workspace limits the classes of task that can be performed.

Multiple robots can be used to accomplish a task, where each one performs its own subtask in parallel, and reduce the production time. This problem often arises in assembly, pick-and-place, and spot welding. When more than one robot operates simultaneously in a common workspace, the problem of avoiding potential collisions between the robots should be considered very carefully.

Corresponding author: Department of Mechanical and Aeronautical Engineering, University of Patras, Rion, Patras 26500, Greece. email: zacharia@mech.upatras.gr

In recent years, considerable attention has been paid to industrial applications [**1**] wherein the robot's end-effector is requested to reach a sequence of task-points exactly once, in an environment cluttered with obstacles, in minimum total cycle time. This problem occurs often in practice, e.g. in spot welding, car painting, and inspection tasks. At each task-point, the robot stops, while its end-effector performs a certain operation.

The successful programming of industrial manipulators and the time spent by the programmer depends on his/her experience and intuition. However, the programmer is not able find the robot motion sequence via the given task-points corresponding to the shortest time, due to the complexity of the searching space. Apart from obstacle avoidance, the programmer thinks in the robot working space, while the robot is controlled in the configuration space. The manipulator moves all its joints simultaneously and its end-effector approaches each task-point with one of the configurations derived by the solution of the inverse kinematic problem.

Therefore, the search space is quite large and it is very difficult and time consuming to program and test alternative routes in a simulator, even for an experienced programmer. To reduce the cycle time, more than one robot could be used for accomplishing point-to-point tasks. It is obvious that, in the case of multiple robots with overlapping workspaces that move simultaneously, it would be far more difficult to program a route with low cycle time. Current robot simulators or virtual manufacturing environments do not include tools for the simultaneous motion planning and scheduling of multiple manipulators. The aim of the present paper is to contribute towards automation of optimal robot programming.

This paper examines the scenario where two robotic manipulators are requested to serve simultaneously a set of task-points cluttered in a three-dimensional (3D) environment. Each manipulator starts from a task-point (initial point), passes through a number of task-points (from each one exactly once), and returns back to its initial point. The objective is to determine the optimum sequence of the task-points that should be visited by each one of the two robots that ensures minimum total cycle time and collision avoidance among their links.

The problem of determining the optimum sequence of a manipulator's task-points in a 3D environment can be considered as a combination of the well-known Travelling Salesman Problem (TSP) [**2**] and the Motion Planning Problem (MPP) [**3**]. Both of these are known to be intractable. To the best of our knowledge, the integration of these problems for a two-robot work cell has not been studied so far.

Several works are published dealing with motion planning of two robots while avoiding collisions among their links, based on different approaches for the solution of the problem. In the work of Nof and Drezner [**4**], the objective was to minimize the total distance travelled by the two robots ignoring collision among their links. It was assumed that all robots are identical and have the same speed. Under this assumption, distance and time are equivalent. The solution tends to be a division of a tour by the user into almost equal parts and assigning each subtour to a robot. Although the two robots work simultaneously, they are located at the two opposite sites of a conveyor and the assembly place is split accordingly, so that no crossing of the robot arms will occur. Under this limitation, collision avoidance is not taken into consideration. The optimization problem is solved using a branch-and-bound technique and then using a heuristic algorithm. The experimental tests using robots with low degrees of freedom exhibit good performance.

Jiang *et al.* [**5**, **6**] presented a scheduling scheme for a two-robot assembly cell. The task-points were partitioned into two groups and are assigned to the robots by the user. To avoid collisions between the cooperating manipulators, it was assumed that only one robot can enter the assembly area at any given time. This constraint was imposed when deriving the assembly-planning algorithm in order to guarantee safe operation. The optimization problem concerns the minimization of the idle time of the robots and the maximization of the utilization of the assembly area. Although conservative, this approach generates near-optimal solutions if the size of the robot end-effectors is comparable to the assembly area.

1.1 Main contribution

'Task scheduling' refers to the computation of the optimum sequence among configurations (points). 'Motion planning' refers to the determination of a collision-free path from one start configuration (point) to a goal configuration (point). The present paper combines two notoriously hard sub-problems: the collision-free shortest path and the task-scheduling problem. It is further complicated by the fact that each goal placement by the end-effector may be achieved by several configurations of the arm (distinct solutions of the arm's inverse kinematics).

The published papers discussed above use methods for planning and scheduling motions for two robotic manipulators by considering the planning and scheduling as two separate problems. This approach has the following drawback and/or limitation: due to the fact that the task-scheduling and motion-planning problem are not simultaneously solved, it cannot be assured that the determined solution is the global best one. In addition, in some cases, the division of the task-points between the two robots is arbitrarily selected by the programmer.

When the two problems are separately solved, the output of the task-scheduling problem is fed as input to the motion-planning problem. However, this approach cannot guarantee a global optimum solution, due to the fact that solving the task-scheduling problem does not include the information of collision-free motion. The configurations at the task-points are collision-free, but who can guarantee that the motion among the task-points is also collision-free? At a second stage, motion planning is solved using the configurations resulting from the task-scheduling problem. Starting from an already limited search space, how possible is it to find a global optimum solution?

Solving the two problems separately, the search is limited; thus, it is rather difficult to come across the global optimum solution. Instead, it is more probable that the provided final solution problem will be far from the optimum one.

This paper presents an integrated approach for two robotic manipulators considering scheduling and planning simultaneously. The advantages of the proposed approach are:

(a) end-effector paths of the robots are generated by taking into account the collision avoidance between the links of the two robots and the multiple solutions of the inverse kinematics;
(b) integration of end-effectors' path planning and task-scheduling planning provides the optimal or near-optimal solution, since the solution is sought all over the search space.

It is considered that the two robots move simultaneously and the division of the task-points between the robots is determined automatically by the searching algorithm towards minimization of the cycle time.

A genetic algorithm (GA) with special encoding is proposed, where the division of the task-points for both robots and the multiple solutions of the inverse kinematics are incorporated. Furthermore, for the collision check between the two robotic manipulators, a sweep line algorithm [**7**] is integrated to the fitness function of the GA.

1.2 Outline of the paper

The rest of the paper is organized as follows. In section 2, the task-scheduling problem and assumptions for a two-robot work cell are defined. Section 3 presents the task-scheduling problem considering the representation of the robotic arm and the satisfaction of the imposed criteria. Section 4 presents the optimization algorithm and section 5 demonstrates and discusses the efficiency of the proposed method through multiple experiments applied for two PUMA 560 robots. Finally, section 6 summarizes the contribution of the paper.

2 PROBLEM FORMULATION AND GENERAL ASSUMPTIONS

In this paper, two robotic manipulators are assumed to be operating in a 3D environment which is cluttered, with a set of task-points that are equally available to both robots. The robotic manipulators are assigned to pass through the task-points in the given environment, so that all the task-points are served by one of the two robots exactly once. A formal statement of the problem under consideration is given in section 2.1.

2.1 Problem statement

Let two robotic manipulators be operating in a 3D environment consisting of a set $TP = \{1, \ldots, p, \ldots, N\} \subset \mathbb{Z}$ of N task-points, as shown in Fig. 1. The numbering of the task-points is arbitrary and the integer number p is the assigned name of this point, with which it appears in the sequence for each robot. Assume that:

(a) the dynamic behaviour of the robots does not affect the solution of the problem;
(b) at each task-point, the robot performs an operation for fixed time, which is not considered in the total cycle time optimization;
(c) each link of the robotic manipulator is represented by a straight-line segment.

Then, the problem addressed in this paper is defined as follows. *Given the anatomy of the two robots and the position of the N task-points, determine a tour for each robotic manipulator satisfying the following criteria and constraints:*

(A) each robot should avoid collision with the other robot;
(B) the robots should visit all task-points, passing through each one of them exactly once in the minimum cycle time;

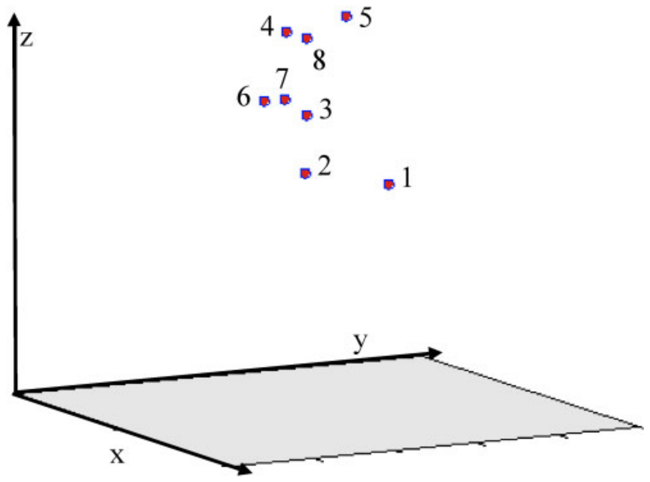

Fig. 1 The set of $TP = \{1, 2, .., 8\}$ task-points in the 3D space

(C) the robots should not visit the same task-points.

Note: *Each β-robot can reach a subset of task-points* $TP_\beta = \left\{p_1^\beta, \ldots, p_i^\beta, \ldots, p_{N_\beta}^\beta\right\} \subset TP$, $\beta = 1,2$, *which contains* $N_\beta, \beta = 1,2$ *task-points. Furthermore, we assume that* $TP = TP_1 \cup TP_2$ *and* $N_1 + N_2 = N$.

2.2 Representation of robotic arm

The proposed approach is general and can be applied to any non-redundant robots. For the sake of simplicity, the performance of the proposed method is investigated in all experiments using two PUMA 560 robots operating in a common 3D environment. A PUMA 560 is an arm with six revolute joints. The first three degrees of freedom (DOFs) are used to take the wrist to a particular position, while the latter three are used to orient the wrist in a desired configuration. Hence, it is the first three DOFs which normally cause the collision. Therefore, only the motion planning of the first three links is considered for collision avoidance. A visual representation of a robotic manipulator is depicted in Fig. 2.

In order to reduce the computational complexity of the problem, each robotic manipulator can be considered as a first-degree B-spline curve $C^\beta(s) = \left(u_1^\beta(s), u_2^\beta(s), u_3^\beta(s)\right)$, $\beta = 1, 2$, which is defined in the 3D normalized environment as follows

$$C^\beta(s) = \sum_{\delta=0}^{4} M_{\delta,1}(s) p_\delta^\beta = \sum_{\delta=0}^{4} M_{\delta,1}(s)\left(x_\delta^\beta, y_\delta^\beta, z_\delta^\beta\right),$$
$$0 \leqslant s \leqslant 1, \ \beta = 1, 2 \qquad (1)$$

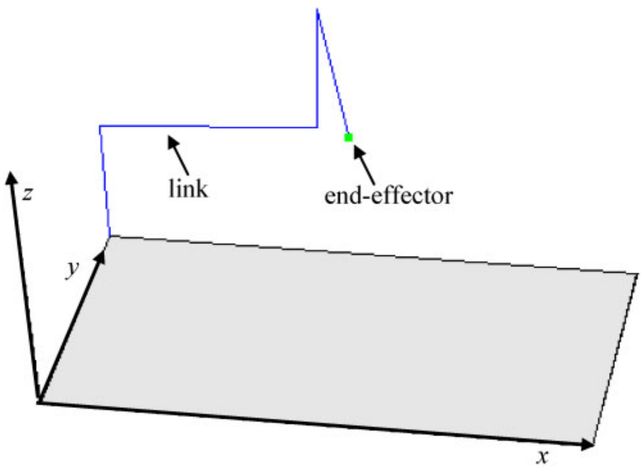

Fig. 2 A PUMA 560 robotic manipulator in the 3D environment

where p_δ^β, $\delta = 0,1,\ldots,4$, $\beta = 1,2$ represents the position of the δ-joint of the β-robot in the 3D space, 4 is the number of joints, and $M_{\delta,1}$ is one-degree base function.

3 TASK SCHEDULING FOR TWO ROBOTIC MANIPULATORS

3.1 Optimum cycle time for robotic manipulator

Various metrics for the evaluation of the tours via given task-points and for searching for the best tour have been proposed in the relevant literature [**8**, **9**]. Some of them were based on the Euclidean norm defined either in the Cartesian or in the configuration space, which corresponds to the distance between two task-points or between two configurations, respectively. These metrics do not take into account the type of motion of the manipulator. The industrial manipulators move their joints simultaneously, so the time for moving from one to another configuration depends on the slowest joint.

A critical issue in industry is the reduction of cycle time; therefore, in this paper the longest time among all joints between two successive configurations is considered as the metric considering that the velocity of each joint is constant. This is a good approximation with minimal effect on the final result since all the trapezoidal velocity profile is approximated by a constant joint velocity.

Let two 6-DOF manipulators operating in the 3D space and have to visit N task-points, the locations of which are known. First, the joint coordinates are computed solving the inverse kinematics problem for each point in the 3D space. Therefore, the time t_β^1 spent by the β-manipulator to travel from the task-point $p_z i^\beta - 1$ using the θ^a set of the joint coordinates, which determines the a-configuration, to the task-point p_i^β using the θ^b set of the joint coordinates, which determines the b-configuration, can be written as

$$t_\beta^1 = \sum_{i=2}^{N_\beta}\left[\max_j\left(\frac{\left|\theta_{ji}^b - \theta_{j(i-1)}^a\right|}{\dot{\theta}_j}\right)\right],$$
$$j = 1, 2, \ldots, 6; \ a, b = 1, 2; \ldots, 8 \qquad (2)$$

where θ_{ji} is the jth joint displacement for the p_i^βth end-effector location and $\dot{\theta}_j$ is the average velocity of joint j. The motion between two successive configurations is designated considering that the joint variables change linearly with time. Equation (2) denotes the fact that the travel time between two positions is determined by the slowest manipulator's joint, which is expressed by 'max'. It is also clear that for the case of multiple solutions, the choice of the manipulator's configuration affects significantly the travel time.

In addition, each β-robotic manipulator should follow a β-tour which starts from an initial task-point, passes through a number of task-points N_β, and returns to the initial task-point, while they should not serve the same work stations. This is expressed by

$$t_\beta^2 = \max_j \left(\frac{\left|\theta_{j1}^b - \theta_{jN_\beta}^a\right|}{\dot{\theta}_j} \right),$$
$$j=1,2,\ldots,6;\ a,b=1,2,\ldots,8 \tag{3}$$

Thus, the total cycle time t_β required by the β-robot to visit N_β task-points and return to the initial task-point is given by

$$t_\beta = t_\beta^1 + t_\beta^2, \quad \beta=1,2 \tag{4}$$

The minimization of the function

$$T=\max(t_1,t_2) \tag{5}$$

with respect to the joint variables $\theta^b \in \Re^6$ gives the minimum cycle time. It should be noticed that the optimum cycle time for each robot depends to a great extent on the multiplicity of the robot configurations corresponding to the task-points.

3.2 Conditions for deriving collision-free motions

As mentioned in section 2.2, each robotic manipulator is modelled as a first-degree B-spline curve defined in the 3D normalized space $[0,1]^3$, where the n-joints are the control points that define the first-degree B-spline curve.

In order to derive collision-free motions for the robotic manipulators, any type of metric could be used between two configurations as an objective function; however, it is well known that these functions mix translational and rotational components [**3**], which are computationally expensive. A common and simple approach is to use a penalty function, where the tours which are not collision-free are excluded from being selected by the GA by assigning them a very big objective value (very low fitness). In this way, these tours have a very small chance to be selected for reproduction in the population of the next generation of the GA. The main disadvantage of this technique is that, in very complex test cases, the GA may not be able to find a solution, even if one exists. Instead, a novel penalized function is proposed in order to derive collision-free motions for the two manipulators as a two-stage process. First, a check for intersections between the two B-spline curves (robotic manipulators) is made by using the sweep line algorithm [**7**]. Then, the exponential function $\exp\left(\sum_{i=1}^{\text{No. of collisions}} ct_i\right)$ (where ct_i is a constant) is constructed, which takes a value in the interval of $(1,+\infty)$ if the two robotic manipulators collide and the value of 1 otherwise. In all of the experiments, the constant ct_i takes a value equal to 5. Extensive experimental results have shown that the proposed penalized function leads always to a solution (collision-free motions), if one exists, and 'helps' the GA to converge to a collision-free tour very quickly.

3.3 Condition for serving requested task-points

Each β-robotic manipulator should follow a tour which eventually connects a subset TP_β of number of task-points N_β with its initial task-point. Since every task-point should be served just once and by only one robotic manipulator, then it holds that

$$TP_1 \cap TP_2 = \varnothing \tag{6}$$

3.4 Overall formulation of task-scheduling problem

Taking the above analysis into consideration, the task scheduling problem defined in section 2.1 is formulated as an optimization problem given by

$$E = \exp\left(\sum_{i=0}^{\text{No. of collisions}} ct_i\right)\cdot T \tag{7}$$
$$\text{subject to } TP_1 \cap TP_2 = \varnothing$$

The minimization of equation (7) with respect to the joint variables $\theta^b \in \Re^6$, $\beta=1,2$ gives a solution satisfying the criteria A to C of section 2.1. It is clear that the complexity of the problem depends on the number of task-points and the number of possible configurations of the manipulators.

4 PROPOSED GENETIC ALGORITHM

GAs [**10**] have been successfully applied to optimization problems with large and complex search spaces due to their ability of reaching a global near-optimal solution even if the search space contains multiple local minima. Besides, GAs are well-suited to complex problems when traditional techniques are too difficult or time consuming to solve. Thus, GAs have been used for the solution of the optimization problem expressed by equation (7).

GAs are probabilistic search methods that employ search techniques inspired by Darwin's theory of evolution. GAs employ a random, yet directed, search for finding the globally optimal solution. In contrast to random sampling algorithms, they have the ability to direct the search towards relatively promising regions in the search.

In this study, a modified GA has been designed and implemented to deal with the optimal task-scheduling problem for two robotic manipulators. The characteristics of the proposed GA are analysed and described in following subsections. Figure 3 presents the mechanics of the proposed GA schematically, where the block 'evolution by GA' involves reproduction, crossover, mutation, and fitness evaluation.

4.1 Initial population

The algorithm starts with an initial random population in order to uniformly distribute the selected chromosomes (solutions over the search space). The result of this run is used to 'seed' the initial population of the next run in the hope of starting the evolution in a more useful region of the search space [11]. 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 [12]. The seeding percentage is set to be 10 per cent of the initial population.

4.2 Chromosome syntax

A key issue in the implementation of the GA is the encoding of the optimization variables; in other words, the encoding of the chromosome. Considering the multiplicity of the inverse kinematics problem for the computation of the total cycle time, each chromosome in the proposed GA consists of three parts. The first part, consisting of integer numbers, represents the sequence of the task-points with which the two robots visit the N task-points. The second part, consisting of N bytes of 3 bits, represents the robot configurations corresponding to each task-point. In other words, each byte of $(000, 001, 010, \ldots, 111)$, corresponding to a task-point in the 3D environment, determines one out of eight configurations of the manipulator [13]. The third part, consisting of $N-2$ binary numbers, determines the number of N_1 task-points corresponding to the 1-robot. The number of task-points N_1 (corresponding to the third part of the chromosome) is an integer number between 1 and $N-1$ and is generated by the number of 1's increased by one. It is taken for granted that N_1 cannot take the values 0 and N, since this would mean that all the task-points

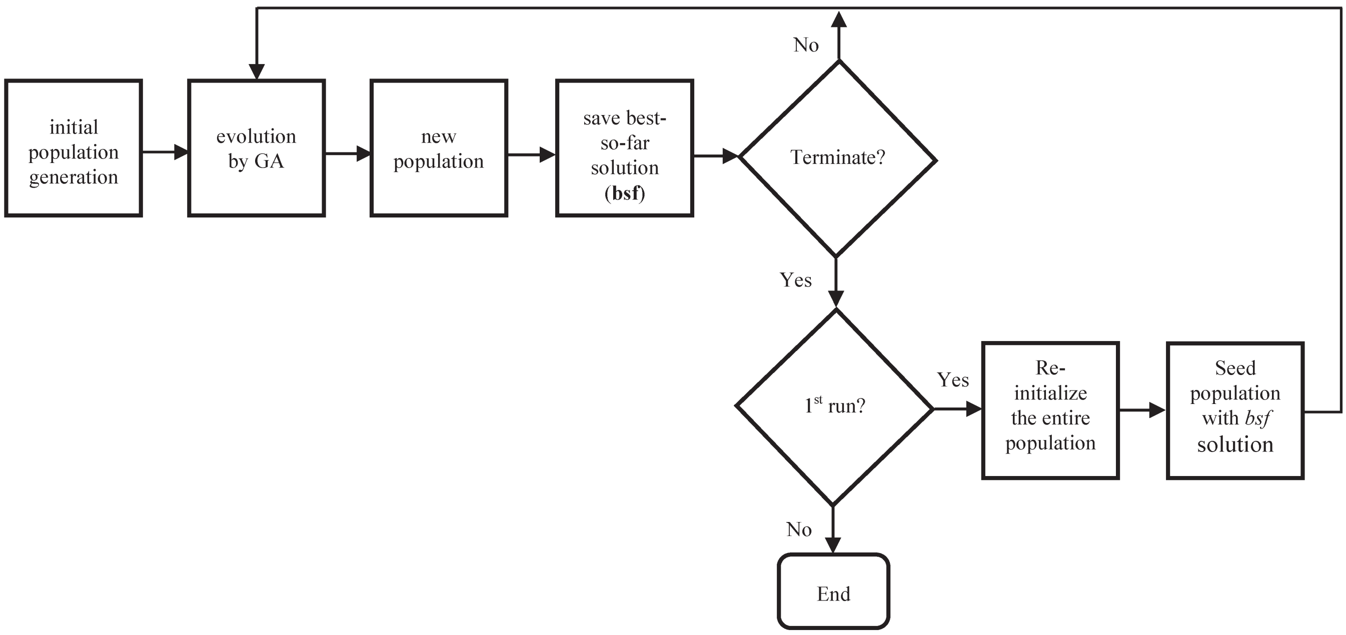
initial population generation
evolution by GA
new population
save best-so-far solution (bsf)
Terminate?
No
Yes
1st run?
Yes
Re-initialize the entire population
Seed population with bsf solution
No
End

Fig. 3 The mechanics of the evolutionary approach proposed

are assigned to only one of the two robots. An example of a chromosome is given below to facilitate the understanding of this special encoding. Let two 6-DOF manipulators have to visit six task-points in the 3D space, then a possible chromosome is in the form of

$$\underbrace{5\ 4\ 3\ 1\ 2\ 6}_{\text{six task-points}}\quad \underbrace{\overbrace{0\ 1\ 1}\ \overbrace{1\ 1\ 0}\ \overbrace{1\ 0\ 1}\ \overbrace{1\ 1\ 1}\ \overbrace{0\ 1\ 0}\ \overbrace{1\ 0\ 1}}_{\text{configurations corresponding to the six task-points}}\quad \underbrace{0\ 1\ 1\ 1}_{\substack{\text{binary string}\\ \text{for the division}}}$$

The randomly generated number $N_1 = 4$ (i.e. three 1's increased by one) determines that the 1-robot is assigned to visit the first four task-points $\{5, 4, 3, 1\}$ with the configuration defined by the genes {011 110 101 111} and the 2-robot is assigned to visit the remaining task-points $\{2, 6\}$ with the configuration defined by the rest genes of the second part of the chromosome.

4.3 Fitness function

The evaluation mechanism (to judge the merit) of all the chromosomes of the population is the fitness function. The value of the fitness function for one chromosome is the reflection of how well this chromosome is adapted to the environment. This indicates the ability of the chromosome to survive and be reproduced in the next generation. The fitness function derives from the objective function as

$$\mathcal{F} = \frac{1}{E} \tag{8}$$

4.4 Genetic operators

The following three genetic operators [**10**, **13**, **14**] were selected for use with the proposed GA.

1. *Reproduction*. Reproduction is a simple copy of an individual from one generation to the next one without any modification, following the process of natural selection and the 'survival of the fittest'. It 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. 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.
2. *Crossover*. Crossover joins together parts of several individuals in order to produce new ones for the next generation. The individuals are randomly selected according to a user-defined probability (crossover rate). The OX crossover is used for the integer part and for the binary part the one cut-point crossover is used.
3. *Mutation*. The inversion operator is used for the integer part of the chromosome, whereas for the binary part of the chromosome the boundary mutation is used. Inversion selects two positions along the string at random and then inverts the subsection of the values between these two positions. Boundary mutation changes the value of a gene with a random number chosen from the permitted range of values.

4.5 Termination conditions

There is not mathematical proof of convergence or any guarantee that the GA will find the global optimum. In the present approach, the algorithm terminates by defining in advance the number of iterations (generations). 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.

5 EXPERIMENTAL RESULTS AND DISCUSSION

The effectiveness of the proposed approach was validated through a number of experiments with two robotic manipulators operating in a common 3D space cluttered with a finite number of task-points. Without loss of generality, the robotic manipulators

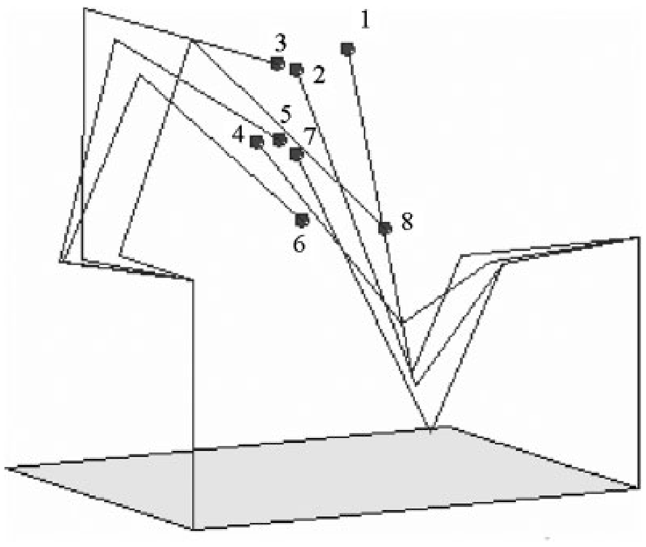

Fig. 4 The resulting robot configurations at the task-points for the first example (test case I)

used for the experiments were identical, i.e. they had the same size and were moving with the same velocity. The overall method was implemented on a Core 2 Duo 2.13 GHz personal computer using MATLAB software. The GA operators were defined after experimentation as follows: population size = 150, maximum number of generations = 400, crossover rate = 0.75, and mutation rate = 0.004. Three experimental tests are presented.

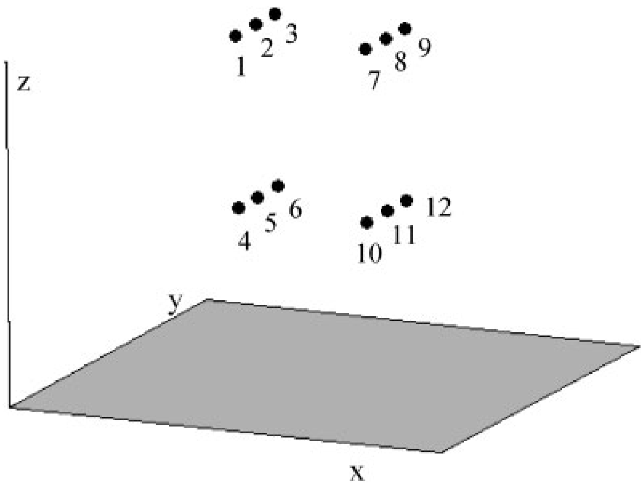

Fig. 6 The 3D environment cluttered with 12 task-points

5.1 Experiments

The first example (test case I) assumes two PUMA 560 robots operating in a 3D environment that should visit $N = 8$ task-points, as illustrated in Fig. 1. The links for both robots are of equal length. Figure 4 depicts the solution of the combinatorial task-scheduling problem for the two robots. As shown in Fig. 5, the robot end-effectors visit the task-points without collisions among their links. The resulting tour for the 1-robot passes from the task-points 5–6–8–3–5 where $N_1 = 4$ and $TP_1 = \{5, 6, 8, 3\}$, while for the 2-robot the resulting tour passes from the task-points 1–2–4–7–1, where $N_2 = 4$ and $TP_2 =$

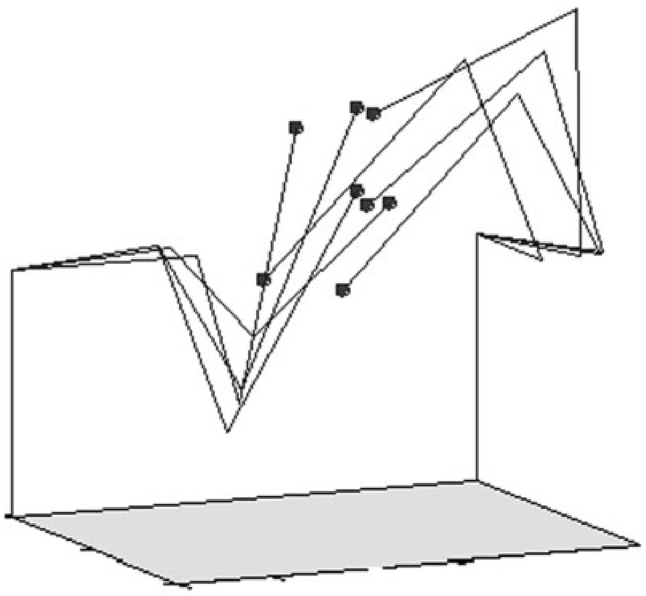

Fig. 5 Another point of view for the resulting solution for the first example (test case I)

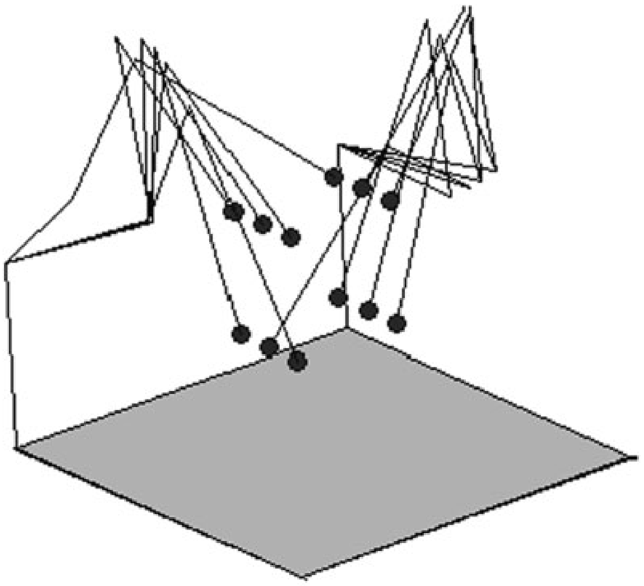

Fig. 7 The resulting robot configurations at the task-points for the second example (test case II)

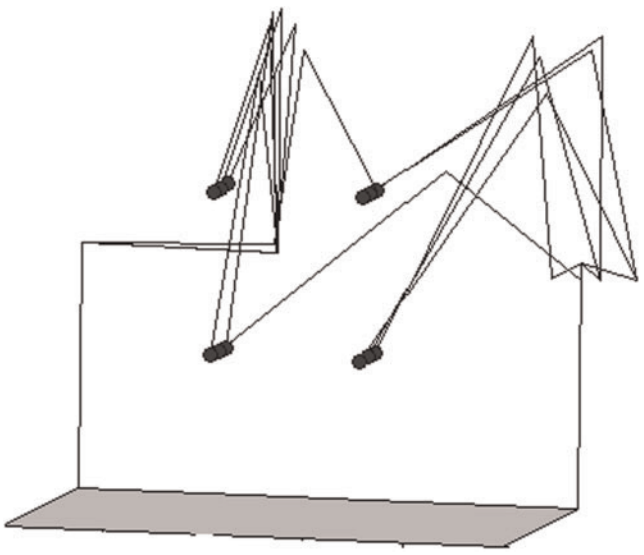

Fig. 8 Another point of view for the resulting solution for the second example (test case II)

$\{1, 2, 4, 7\}$. As a result, the eight task-points are equally distributed to the two robots in order that the total cycle time is minimized.

In the second example (test case II), two PUMA 560 robots operate in a 3D environment and are assigned to visit $N = 12$ task-points, shown in Fig. 6. Figure 7 depicts the solution of the task-scheduling problem for the two robots. Figure 8 shows another point of view for the resulting solution to illustrate that the robot end-effectors visit the task-points without collisions among their links.

The resulting tour for the 1-robot passes from the task-points 9–3–2–1–4–6–9, where $N_1 = 6$ and $TP_1 = \{9, 3, 2, 1, 4, 6\}$, while for the 2-robot the resulting tour passes from the task-points 5–10–11–12–8–7–5, where $N_2 = 6$ and $TP_2 = \{5, 10, 11, 12, 8, 7\}$.

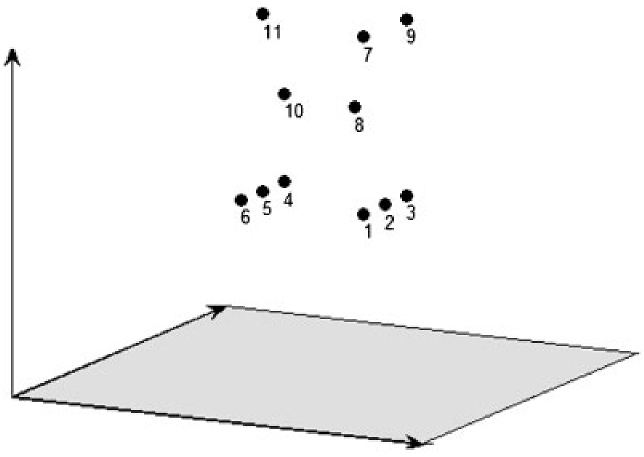

Fig. 9 The 3D environment cluttered with 11 task-points

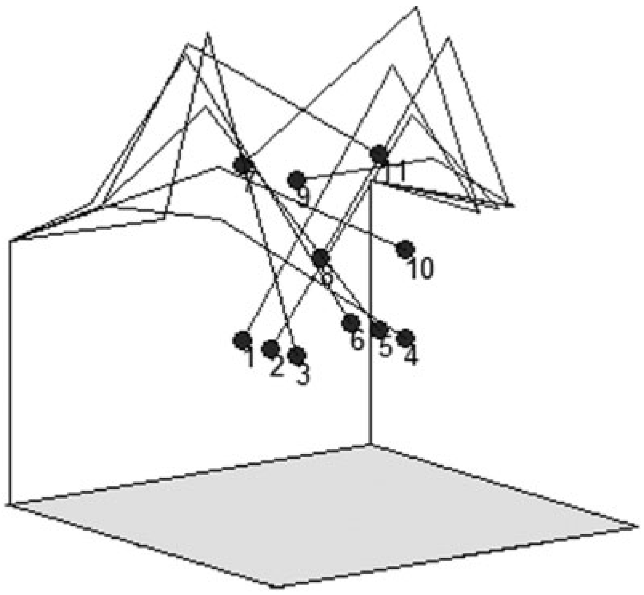

Fig. 10 The resulting robot configurations at the task-points for the third example (test case III)

In the third example (test case III), it is assumed that two robots operate in a 3D environment cluttered with $N = 11$ task-points, as shown in Fig. 9. Figure 10 depicts the solution of the task-scheduling problem for the two robotic manipulators. Figure 11 shows another point of view for the resulting solution to illustrate that the robot end-effectors visit the task-points without collisions among their links.

The solution tour for the 1-robot passes from the task-points 1–7–9–8–2–1, while for the 2-robot the resulting solution passes from the task-points 6–3–5–4–10–11–6.

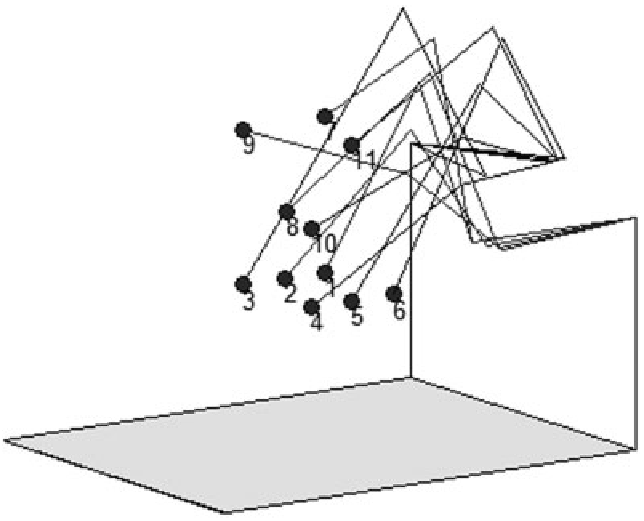

Fig. 11 Another point of view for the resulting solution for the third example (test case III)

Table 1 Comparison between the proposed and the standard method

			Test case I	Test case II	Test case III

Proposed method		Task-points	Robot 1: {5, 6, 8, 3, 5}
			Robot 2: {1, 2, 4, 7, 1}
				Robot 1: {9, 3, 2, 1, 4, 6, 9}
				Robot 2: {5, 10, 11, 12, 8, 7, 5}
					Robot 1: {1, 7, 9, 8, 2, 1}
					Robot 2: {6, 3, 5, 4, 10, 11, 6}

		Total cycle time (s)	13.7	16.1	15.4

Two-phase method	Task scheduling	Task-points	Robot 1: {3, 2, 7, 5, 4}
			Robot 2: {1, 8, 6}
				Robot 1: {1, 2, 3, 6, 5, 4}
				Robot 2: {9, 8, 7, 10, 11, 12}
					Robot 1: {11, 10, 4, 5, 6 }
					Robot 2: {9, 7, 8, 1, 2, 3}

		Cycle time (s)	9.4	12.9	11.6
	Path planning	Cycle time (s)	5.8	7.2	6.2
		Total cycle time (s)	15.2	20.1	17.8

In two of the examples presented, the task-points are equally distributed to the two robots as shown in test cases I and II. However, this is not a rule, since the result depends on the position of the task-points, the orientation with which the robot is assigned to reach the task-points, and the positions of the robot bases. Thus, one of the two robots may be assigned more task-points than the other in another scheme (test case III). Furthermore, one should bear in mind that the optimization is in terms of time determined in the joint-space and not the distance between the points in the Cartesian space. Therefore, intuition may delude, since optimization in time does not necessarily conform to optimization in Cartesian distance. Comparing the total cycle time corresponding to the resulting optimum solution of test case II, which is 16.1 s, with the total cycle time corresponding to the 'optimum' sequence by intuition (for the 1-robot: 3–2–1–4–5–6–3 and for the 2-robot: 9–8–7–10–11–12–9), which is 21.3 s, the conclusion is that near-optimum cycle time is derived from the algorithm.

It is difficult to compare different techniques because they were tested on different types of environment, using different underlying libraries, and implemented on different machines. However, one can compare the proposed method with a two-phase algorithm which considers the two tasks (task scheduling and path planning) separately. The results of the comparison are presented in Table 1.

5.2 Computational time study

This subsection refers to the study of the required time for the solution of the time-optimum task-scheduling problem with varying number of task-points. An experiment was carried out in a 3D environment cluttered with two robotic manipulators with varying task-points in the range of eight and 18. Figure 12 depicts the computational time of the central processing unit (CPU) versus the number of task-points, where it seems that the required CPU time is almost linear.

6 CONCLUSIONS

In this work, a new method is introduced for determination of the (near-)optimum sequence of task-points for two articulated robots, operating in a

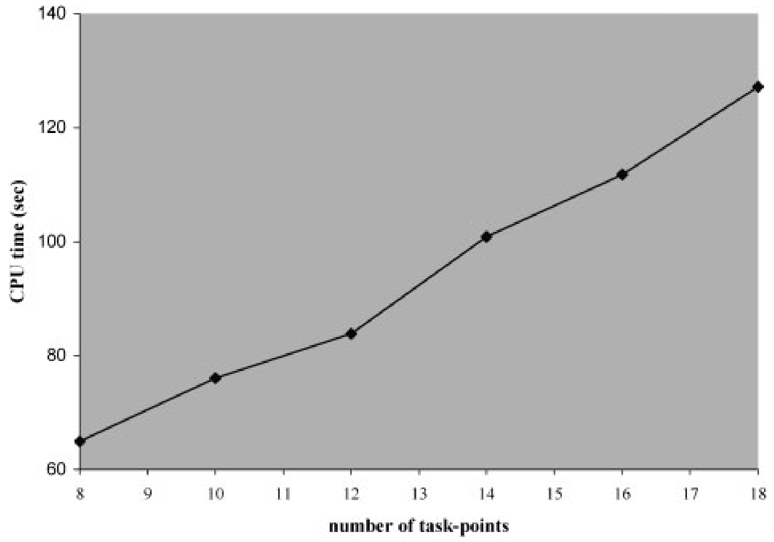

Fig. 12 CPU time graph

common 3D environment, which ensures collision avoidance at the task-points. The proposed approach is based on a special encoding of the GA chromosome to take into account the multiple considerations resulting from the solution of inverse kinematics.

The experimental results demonstrate that the algorithm is capable of determining the (near-)optimum sequence of task-points for both robotic manipulators operating in the same 3D environment by considering the multiplicity of robot configurations, while ensuring collision avoidance between the robot links at the task-points.

Future work will concentrate on applying the proposed concept in more complicated 3D environments cluttered with static and/or dynamic obstacles and including more than two robotic manipulators, for applications in real-world cells.

REFERENCES

1 **Nakamura, Y.** *Advanced robotics, redundancy and optimization*, 1991 (Addison-Wesley, Reading, Massachusetts, USA).

2 **Lawer, E., Lenstra, J., Rinnooy Kan, A.,** and **Shmoys, D.** *The travelling salesman problem*, 1985 (John Wiley, Chichester, UK).

3 **Latombe, J. C.** *Robot motion planning*, 1991 (Kluwer Academic Publishers, Boston, Massachusetts, USA).

4 **Nof, S. Y.** and **Drezner, Z.** The multiple-robot assembly plan problem. *J. Intell. Robot. Syst.*, 1993, **5**(1), 57–71.

5 **Jiang, K., Seneviratne, L. D.,** and **Earles, S. W. E.** Assembly scheduling for an integrated two-robot workcell. *Robot. Comput. Integrated Mfg*, 1997, **13**(2), 131–143.

6 **Jiang, K., Seneviratne, L. D.,** and **Earles, S. W. E.** Scheduling and compression for a multiple robot assembly workcell. *Prod. Plann. Contr.*, 1998, **9**(2), 143–154.

7 **Preparata, F. P.** and **Shamos, M. I.** *Computational geometry*, 1985 (Springer-Verlag Inc., New York, New York, USA).

8 **Wurll, C.** and **Henrich, D.** Point-to-point and multi-goal path planning for industrial robots. In Special Issue on Advances in Practical Motion Planning. *J. Robot. Syst.*, 2003, **18**(8), 445–461.

9 **Saha, M., Roughgarden, T., Latombe, J.-C.,** and **Sánchez-Ante, G.** Planning tours of robotic arms among partitioned goals. *Int. J. Robot. Res.*, 2006, **25**(3), 207–223.

10 **Goldberg, D. E.** *Genetic algorithm in search, optimization and machine learning*, 1989 (Addison-Wesley, Reading, Massachusetts, USA).

11 **Davis, L.** Applying adaptive algorithms to epistatic domains. In Proceedings of 9th the International Joint Conference on Artificial Intelligence, Los Angeles, CA, August 1985, pp. 162–164.

12 **Thomsen, R., Fogel, G.,** and **Krink, T.** A clustal alignment improver using evolutionary algorithms. In Proceedings of the Fourth Congress on Evolutionary Computation – CEC'02, 2002, vol. 1, pp. 121–126.

13 **Zacharia, P.** and **Aspragathos, N.** Optimal robot task scheduling based on genetic algorithms. *Robot. Comput. Integrated Mfg*, 2005, **21**(1), 67–79.

14 **Falkenauer, E.** *Genetic algorithms and grouping problems*, 1998 (John Wiley and Sons Ltd, Chichester, UK).

15 **Michalewitz, Z.** *Genetic algorithms + data structures = evolution programs*, edition 3, 1996 (Springer-Verlag Inc., New York, New York, USA).