Robotics and Computer-Integrated Manufacturing 34 (2015) 140–150

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Time scheduling and optimization of industrial robotized tasks based on genetic algorithms

Khelifa Baizid [a,*], Ali Yousnadj [b], Amal Meddahi [a], Ryad Chellali [c], Jamshed Iqbal [d]

[a] Department of Electrical and Information Engineering, University of Cassino and Southern Lazio, Via G. Marconi, 10, Cassino, Italy
[b] Laboratory of Structure Mechanics, Polytechnics Military School, BP 17, Bourdj-El-Bahri, 16111 Algiers, Algeria
[c] PAVIS, Istituto Italiano di Tecnologia, Via Morego 30, Genova 16163, Italy
[d] Department of Electrical Engineering, COMSATS Institute of Information Technology, Park Road, Chak Shahzad, Islamabad, Pakistan

ARTICLE INFO

Article history:
Received 28 June 2014
Received in revised form
10 December 2014
Accepted 27 December 2014
Available online 22 January 2015

Keywords:
Task time optimization
Genetics algorithms
Industrial manipulators

ABSTRACT

Today's industrial manipulators are more and more demanding in terms of productivity. This goal could be achieved by increasing speed of the robot manipulator and/or by optimizing the trajectories followed by manipulators while performing manufacturing, assembling, welding or similar tasks. Focusing on the second aspect, this research proposes a method based on genetic algorithms by exploiting CAD (computer aided design) capabilities to optimize and simulate cycle time in performing classical manufacturing tasks. The goal is to determine the shortest distance traveled by the robot manipulator in the coordinate space for every pair of successive points. In addition, our optimization procedure considers supplementary factors such as inverse kinematic model (IKM) and relative position/orientation of the manipulator w.r.t task points. All these factors were statistically assessed to determine both individual and cross influences in finding the optimal solution. The proposed approach has been validated on a real life setup, involving a 6-DOFs (degrees of freedom) industrial robot manipulator when performing a spot welding task on a car body. The obtained results are promising and show the effectiveness of the proposed strategy.

1. Introduction

In manufacturing plants [1], several features of industrial manipulators such as flexibility, versatility and adaptability help to accomplish tasks within large environment variations and make them adaptable for various assignments [2]. Moreover, they must work as fast as possible in order to increase productivity thereby decreasing the production costs [2]. Often, researches on trajectory optimization of robot manipulators focus on repetitive tasks such as spot-welding, laser and wafer cutting, handling parts and many other applications [1,3–6], where the high productivity can be obtained by finding a minimum time of execution. Taking into consideration the complexity of the manufacturing systems, the performance of industrial manipulators can be improved by using optimization techniques [7,8] and off-line programming process [10] that drives the end product to have high quality and low cost. Targeting the same goals, various tools and methods have been consequently developed [11]. One of the earliest works on time optimization was presented by Khan and Roth in [12] by using linearization techniques. Later, Bobrow et al. and Shin and McKay proposed an efficient algorithm for finding the optimal trajectory based on the possibility of parameterizing the path with a single scalar variable [7,13]. Dubowsky and Blubaugh [14] considered point-to-point tasks using a method combining simple time-optimal motions. Authors in [4] implemented an algorithm for fruit-picking robot manipulator based on the nearest neighbor (NN) to obtain the near-optimal time path between the fruit locations without taking into consideration the possibility of collision; also the IKM (inverse kinematic model) configurations that were used to determine the optimal sequence were not mentioned. Another method based on the potential of elastic-net was used by Petiot et al. in [15] to minimize the tasks' time cycles. The algorithm schedules the trajectory points in such a way that gives minimum time, unfortunately, the method handles only 2 to 3-DOFs (degrees of freedom) systems because of computational costs. GA is one of the successful methods that retains the ability to search in the midst of large complex spaces [5,16,17]. Usually for a given task, a manipulator visits all the task-points and returns to its initial configuration. In many operational scenarios, the order of visiting the task-points has no effect on accomplishment of required task e.g., point-to-point tasks [14]. However, the total cycle time

* Corresponding author.
E-mail address: baizid.khelifa@gmail.com (K. Baizid).

http://dx.doi.org/10.1016/j.rcim.2014.12.003
0736-5845/

needed to visit all these points can be a function of visit order since cycle time is strongly related to distance traveled by a manipulator. So, this last can perform a task in shorter time if the task trajectory is shorter. This is analogous to our daily life where peoples travel from one place to another and the traveling time corresponds to the distance covered. Also, this is very similar to traveling salesman problem (TSP) [18], where a salesman has to visit a pre-defined number of cities starting and ending at the same location. The objective is to search the minimum traveling tour (to reach all cities) by considering the distances among various cities during the search process. Consequently, the problem of minimizing cycle time of a robotized task can be addressed by determining manipulator's displacement required to visit task-points and return to the initial configuration. However, in this case the problem is more complicated since the traveling sequence needed to achieve a task is performed in joints space rather than in operational space. Furthermore, other factors such as position and orientation of a manipulator, with reference to the task-points, can affect cycle time. Inspired from TSP, researchers have optimized the robotized tasks using genetic algorithms (GAs) [19–21]. The objective function of the optimization problem was calculated based on the path traveled by a robot manipulator in operational space and average joints velocity. As reported, these methods have good impact even with wide task-points space. However, the mentioned approaches do not consider multiple configurations of a robot manipulator. Moreover, this research has been limited to manipulators of 2–3 DOFs. Zacharia and Aspragathos have presented generalized methods for robot manipulators with more than 3*DOFs* [5]. This research has also taken into account the multiplicity of the IKM of a manipulator. The same approach has been recently extended by considering obstacle avoidance [22]. A recent example of research inspired by TSP has been reported in [23] that considers task with continuous trajectories. Most of these methods do not consider a manipulator's placement and orientation, which has a significant influence on total cycle time as was shown in [9,24]. A manipulator can reach each task point by several IKM configurations and from multiple possible placement locations and orientation angles. One of these configurations can be achieved in less time in comparison with others. In [17], authors addressed this problem by taking in addition a manipulator's placement to improve the results. The placement zones have been determined graphically and cycle time has been computed at each point of this zone. However, this method exhibits limitations since it does not consider the orientation of manipulator w.r.t. task-points. The proposed method in the present research is aimed at addressing the constraints mentioned above and it can be applied to point-to-point as well as continue trajectories. So, this is a challenging problem which deals with higher searching space and its solution demands consideration of following four parts: the first one deals with sequence of task-points that a manipulator's end-effector must visit; the second aspect represents manipulator's IKM configurations at each task-point; the third and the fourth parts, respectively, represent the relative placement and the relative orientation between the manipulator base frame and the task-points. The optimization problem is to find the best combinations of these parts that gives an optimal or near optimal cycle time. By taking various combinations from these four parts, importance and influence of each part on the total cycle time of the algorithm performance has been investigated in this research. After optimizing the robotics task, we validate the found solution using a graphical engine that was developed based on SolidWorks® and an API (application programming interface) application under the visual basic for application (VBA). It is worth noting that the collision is resolved by intermediate points defined through the CAD (computer aided design) model of the robotized site. This paper is organized as follows: Section 2 formulates the problem. Section 3 presents the proposed optimization approach. Section 4 describes the simulation setup developed to access the proposed approach. Results of the optimization process and graphical validation are discussed in Section 5. Finally Section 6 concludes the paper.

2. Problem formulation

In robotized manufacturing plants, the time required to perform a given task (cycle time) depends on the traveled distance by the manipulator, which is related to the sequence of the task-points visited by the end-effector (task-points order). On the other hand, the manipulator executes the task in the *operational space* and performs the motion in the *coordinate space*, which makes the traveled distance depending from the manipulator's IKM. Thus, the optimization problem is concerned with finding the minimum distance between each two consecutive task-points. Moreover, the IKM is strongly affected by the placement and the orientation of the manipulator. Therefore, in order to have a complete optimization method (with comparison to those proposed earlier such as [5,17]) we proposed an algorithm that incorporates in its objective function the order of achievement of the task-points, the IKM at each task-point, the relative placement and orientation between the manipulator and the task-points. Similar to [17,5], cycle time has been computed based on a manipulator joints' displacement between each successive pair of points and the average velocity of the corresponding joint.

Consider *n-DOFs* manipulator shown in Fig. 1. It has to visit N points, which represent the required task in six-dimensional space (3 positions and 3 orientations) and return to its initial configuration after task accomplishment. The relationship between joint space and operational space can be formulated as in the following equation:

$$P(t) = f(q(t)) \tag{1}$$

where $q(t) \in R^n$ is the vector of joint coordinates, n is the number of DOFs of a manipulator, $P(t) \in R^N$ is the path to be followed in operational space and f denotes the manipulator IKM [25]. It is worth mentioning that the IKM solution at a given task-point i exists if $P(i) \in \Gamma$, where Γ represents the workspace of a manipulator. The manipulator can reach each task-point with m different configurations from multiple possible placement locations and orientation angles. Consequently, the objective is to find the specific sequence to visit all task-points that gives the smallest cycle time; considering the four parts mentioned earlier.

At a given position and orientation of the manipulator $\Theta(x, y, z, \theta, \psi, \phi)$, the time t_i^{Θ} required by the manipulator to travel

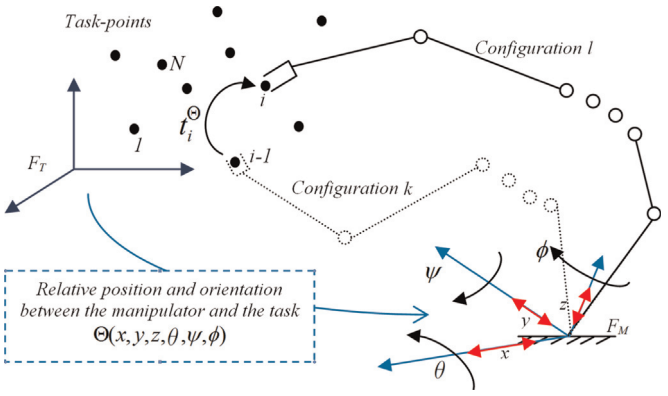

Fig. 1. Schema of *n-DOFs* manipulator that visits N points.

from point $i-1$ using kth IKM configuration to point i using lth IKM configuration is defined by the following equation:

$$t_i^{\Theta} = \max\left(\frac{\left|q_{j(i)}^{l} - q_{j(i-1)}^{k}\right|}{\dot{q}_j}\right)_{\Theta}, \quad j = 1, 2, \ldots, n \tag{2}$$

where $k = 1, 2, \ldots, m$ denotes the possible IKM configurations at task-point $i-1$, $l = 1, 2, \ldots, m$ denotes the possible IKM configurations at task-point i, $q_{j(i)}^{k}$ and $q_{j(i)}^{l}$ are displacements of joint j at task-point $i-1$ and i respectively, Θ denotes the relative placement and orientation of the manipulator w.r.t. task-points, which used to determine the transformation from task-points frame F_T to manipulator fixed frame F_M, $\dot{q}_j$ is average velocity of jth joint. The total cycle time required for a given task t_{cycle}^{Θ} is given by the following equation:

$$t_{cycle}^{\Theta} = \sum_{i=1}^{N+1} t_i^{\Theta} \tag{3}$$

where $N+1$ represents the number of displacements traveled between each two successive configurations plus the return to the initial configuration (i.e., the initial configuration is independent from the task-points). Consequently, the overall formulation function of the optimized task's time can be written as

$$t_{min}^{\Theta} = \min \sum_{i=1}^{N+1} t_i^{\Theta}\left(j, i, k, l, q, N, \Theta(x, y, z, \theta, \psi, \phi)\right) \tag{4}$$

The computation of minimum cycle time t_{min}^{Θ} given by (4) demands carefully examining all the possible solutions, which are in function of all variables of t_i^{Θ}. Consequently, this may significantly increase the complexity because the number of associated solutions is very huge, as multiple parameters construct our objective function. The number of IKM configurations for a manipulator with 6 DOFs is 2^4 [26]. This can be reduced to 2^3 for manipulators having the last three joints axes intersected [25,27]. Therefore, the possible IKM configurations at each task-point can be defined as 2^d where $d = 1, 2, 3, 4$. This number can be increased to $(2^d)^N$ for considering all task-points. Additionally, the possible number of task-points order is defined by $N!$, which can be reduced to half considering symmetry of the problem e.g., the sequence of visiting the points 1–2–3 is same as 3–2–1. For example, the number of placements and orientations of the manipulator w.r.t. task-points is related to the number of grids (*divisions*) that construct the placement and orientation zones [24]. In our approach, the computations are based on 3*D* models of task-points and the manipulator in addition to its workspace as described in [24]. We define two positive numbers ξ_1 and ξ_2 to denote possible placements and orientations, respectively. Their overall values are given by the following equations:

$$\xi_1 = x_{node} \times y_{node} \times z_{node} \tag{5}$$

$$\xi_2 = \theta_{node} \times \psi_{node} \times \phi_{node} \tag{6}$$

where $*_{node}$ denote the number chosen to dived the placement/orientation zone on the corresponding axis.

Considering all the mentioned parameters, the total number of possible solutions that construct our search space is given by the following equation:

$$S = N!/2_{*}(2^d)^N \times \xi_1 \times \xi_2 \tag{7}$$

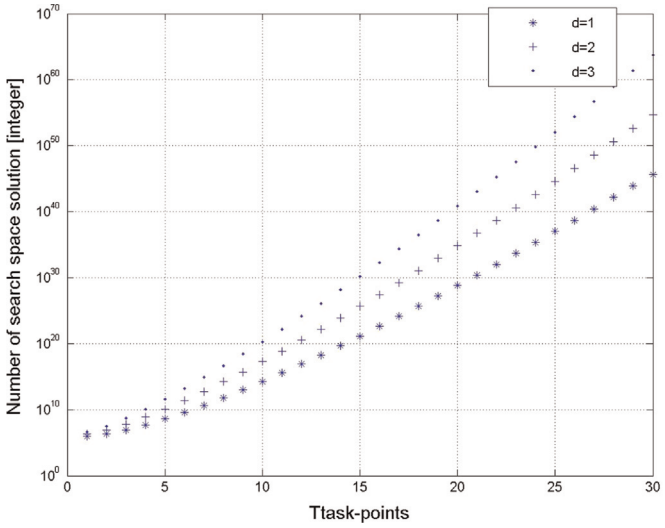

Fig. 2. Number of search space solutions as a function of task-points with varying IKM configurations and constant number of placements and orientations.

2.1. Computational time

The search space in our problem is significantly high as can be seen in Fig. 2, which shows the number of solutions corresponding to $d = 1, 2, 3$, $N = 1, 2, \ldots, 30$ and $\xi_1 = \xi_2 = 1000$. To examine all possible solutions may be practically realistic in case of a low *DOFs* manipulator and with relatively less number of task-points. However, the computational time increases directly with the number of IKM configurations and the number of task-points in almost exponential fashion. Table 1 shows the time, with *Minutes*, required to probe all solutions (with $N=10$) using an application developed under a VBA and running on a 2 GHz computer.

2.2. Example of 2–DOFs manipulator

To illustrate the objective function of the proposed approach, an example of a 2-*DOFs* planar manipulator is illustrated in Fig. 3. The manipulator visits three points *A*, *B* and C in operational space and can perform a task with six different positions and three different orientations and by two different IKM configurations at each task-point. Suppose that the manipulator is currently at position number 5 (located on (2, 2) in *x* and *y* axes, respectively) and on orientation number 2 around *z*-axis, and it uses the 2nd and the 1st configurations to visit A and B–C, respectively. The traveling times from the initial configuration *I* to *A*, *A* to *B*, then *B* to *C* and finally *C* to *I* are given by the following equations, respectively

$$t_1^{5,2} = \max\left(\frac{|q_{(1,A)}^{1} - q_{(1,I)}^{2}|}{\dot{q}_1}, \frac{|q_{(2,A)}^{1} - q_{(2,I)}^{2}|}{\dot{q}_2}\right)_{5,2} \tag{8}$$

$$t_2^{5,2} = \max\left(\frac{|q_{(1,B)}^{1} - q_{(1,A)}^{2}|}{\dot{q}_1}, \frac{|q_{(2,B)}^{1} - q_{(2,A)}^{2}|}{\dot{q}_2}\right)_{5,2} \tag{9}$$

Table 1
Computational time required to scan all possible solutions to find the optimal task's time: T (task-points visit order), C (configurations), P (placement), O (orientation).

Parameter	T	C	P	O
T	T: 1.05e+2	TC: 1.13e+11	TCP: 1.13e+14	TCPO: 1.13e+17
C		C: 6.26e+4	CP: 6.26e+7	CPO: 6.26e+10
P	TP: 1.058e+5		P: 0.0583	PO: 58.3333
O	TO: 1.058e+5			O: 0.0583

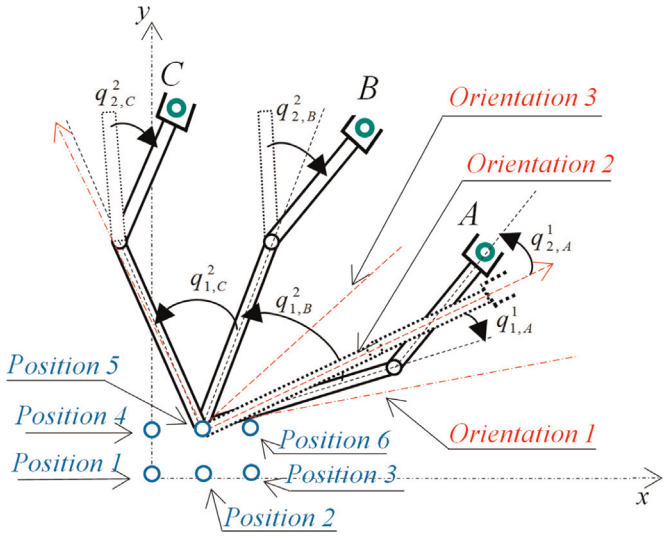

Fig. 3. 2-*DOFs* planar manipulator visiting three task-points, *A*, *B* and *C* in 2*D* operational space.

$$t_3^{5,2} = \max\left(\frac{|q^1_{(1,C)} - q^2_{(1,B)}|}{\dot{q}_1}, \frac{|q^1_{(2,C)} - q^2_{(2,B)}|}{\dot{q}_2}\right)_{5,2} \quad (10)$$

$$t_4^{5,2} = \max\left(\frac{|q^1_{(1,I)} - q^2_{(1,C)}|}{\dot{q}_1}, \frac{|q^1_{(2,I)} - q^2_{(2,C)}|}{\dot{q}_2}\right)_{5,2} \quad (11)$$

where q_I^j denotes the initial position values of the joints $j=1,2$ at the configuration *I*. The overall time required to accomplish the complete task can be expressed as

$$t^{5,1}_{cycle} = t_1^{5,1} + t_2^{5,1} + t_3^{5,1} + t_4^{5,1} \quad (12)$$

Since we are not sure if Eq. (12) represents the optimal time, hence all possible solutions of the problem need to be examined. Therefore, the total number of possible task-points order in this example is $3!/2 = 3$ which are ABC, ACB and BAC. The total number of possible IKM configurations is $(2^1)^3$ which are 222, 221, 212, 211, 122, 121, 112 and 111 while the total number of possible placements and orientations are $3 \times 2 \times 1 = 6$ and $1 \times 1 \times 3 = 3$. Therefore, the total number of solutions is $S = 3 \times 8 \times 6 \times 3 = 432$, so, the problem of finding optimal solution necessitates analysis of all these 432 possibilities.

2.3. Example of 6-DOFs manipulator

In this example, the influence of parts (task-points order, IKM configuration, placement and orientation) on cycle time is investigated. The number of task-points, performed by a 6-DOFs industrial manipulator (Stäubli RX-130 XL), is chosen as 12.

2.3.1. Influence of the task-points order

Given that a manipulator can execute a task in a variety of sequences, where one of them is optimal, the effect of task-points order on the value of cycle time is analyzed. One such typical case is demonstrated in Fig. 4, which shows that the difference between sequence 1, 2, …, 6 and sequence 1, 11, …, 8 is 4.0096 s. This noticeable difference is obtained by changing only almost half of the points sequence (i.e., points 1, 3, 4, 9, 7 are same in both cases).

2.3.2. Influence of IKM configurations

To show the effect of the manipulator's kinematics on cycle time, an example showing two typical sequences is illustrated in Fig. 5. The difference between sequence 2, 3, 4, …, 4 and sequence 3, 2, 4, …, 1, which is very significant as can be seen from the corresponding values of t_c. The sensitivity in value of cycle time even by variation of a single configuration is evident at point 8 (red sequence), where changing configuration from 3 to 1 resulted in increase of cycle time from 50.9893 s to 52.5389 s. Changing multiple configurations makes this difference dramatically even higher. It is worth noticing that the difference of cycle time in various configurations is related to a known number of task-points. In case this number is higher, the difference of cycle time may be even more significant.

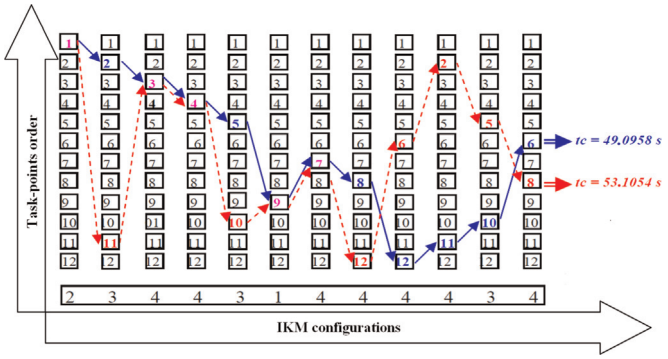

Fig. 4. Time of task accomplishment corresponding to two typical task-points order.

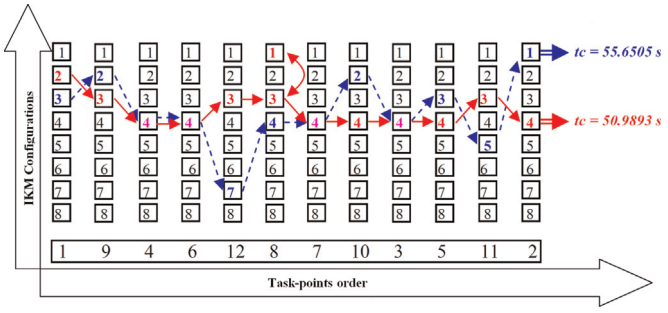

Fig. 5. Two sequences of IKM configurations according to one sequence task-points order. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

2.3.3. Influence of the placement zone

We change the placement the manipulator along the three axes (*X*, *Y* and *Z*) and evaluate cycle time, the manipulator placement zone is divided into 100 divisions (nodes) on each axis. The task-points order and IKM configurations are known. Evaluating time cycle on all positions, Fig. 6 shows scans of placement zones on *XY*, *XZ* and *YZ* planes. A general remark on these three graphs is that the variation of task cycle time for each manipulator's position is completely a stochastic function. This phenomenon is comparatively more obvious in *YZ* plane (Fig. 6(c)), where task-points are more widely distributed from geometrical point of view. The position area (in term of nodes) that gives less time in XY plane is located from 45 to 58 and from 90 to 100 on *X* and *Y* axes, respectively. Similar results have been observed in *YZ* plane (from 80 to 95 and from 59 to 100 on *Y* and *Z* axes, respectively). Observing these surface plots for minimum cycle time, it is found that on *XY* plane, this time is 61.09 s at the position of 52, 91, and 01 nodes (on *X*, *Y* and *Z*, respectively). The cycle times in *XZ* and *YZ* planes are found to be 61.25 s at 52, 39, and 01 and 59.32 s at 04, 04, and 73 position (on X, Y and Z, respectively).

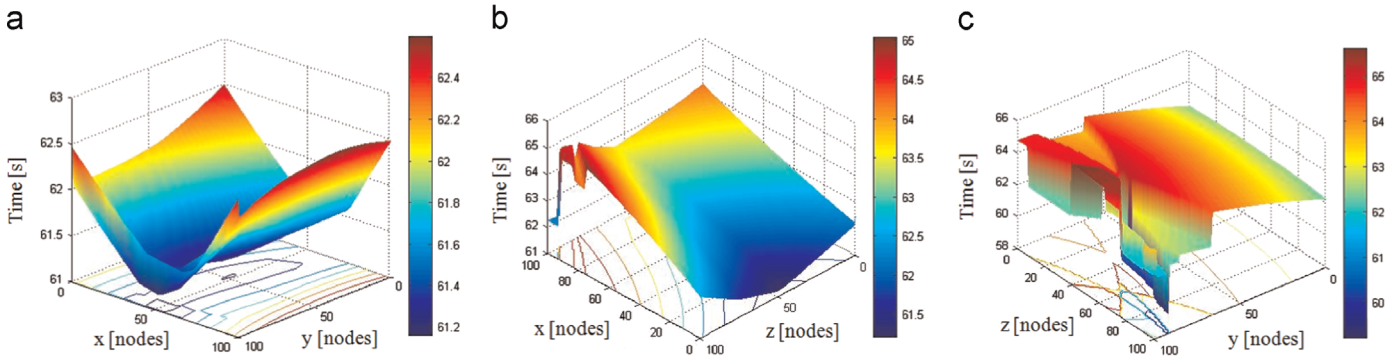

Fig. 6. Cycle time by scanning the placement zone (a) manipulator's position along *X* and *Y*, (b) position along *X* and *Z*, and (c) position along *Y* and *Z*.

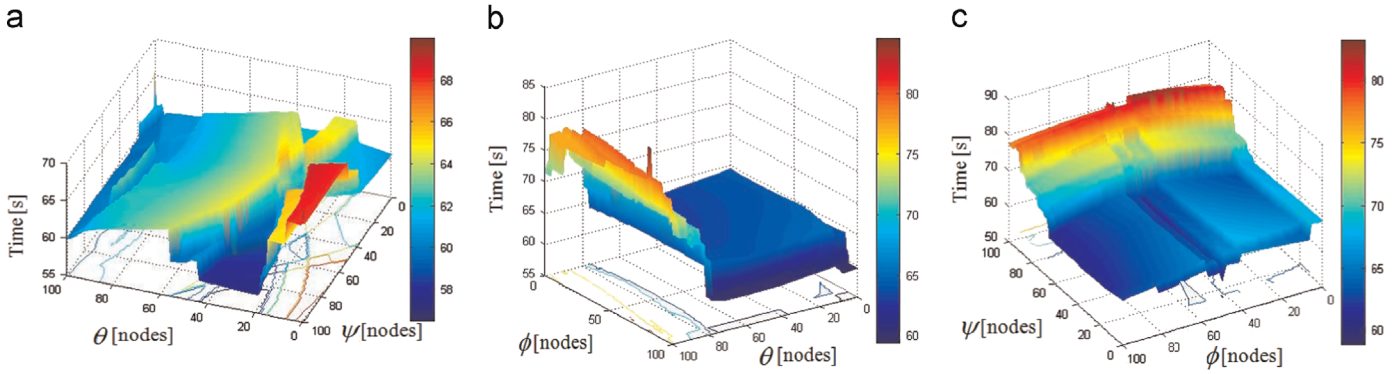

Fig. 7. Cycle time by scanning the orientation zone: (a) orientation around *X* and *Y*, (b) orientation around *X* and *Z*, and (c) orientation around *Y* and *Z*.

2.3.4. *Influence of the orientation zone*

The orientation zone has been scanned w.r.t. three angles (θ, ψ and ϕ) by fixing one of them and varying the others as shown in Fig. 7(a)–(c). Observing these three figures, which are combinations of two by two angles, the optimum task time has been found to be 57.48 s corresponding to orientation of 80, 38, and 69 nodes on θ, ψ and ϕ, respectively. However, scan of the whole orientation zone leads to find a new optimal value of 56.98 s in 97, 85, and 28 nodes.

3. GAs optimization approach

GAs are stochastic optimization methods proposed by John Holland based on the Darwinian evolution theory [28]. The idea behind GAs is artificial evolution of a population of chromosomes which represent the possible solutions to the concrete problem [29]. The initial population is generated randomly. It then gets evolved through several operations within a defined number of generations. During every generation, each chromosome of the population is evaluated according to its objective function w.r.t. specific conditions followed by selection of best chromosomes to participate in reproduction of the next generation. The reproduction procedure generates new offspring based on the genetic material of two parent chromosomes, by applying crossover and mutation operations with a certain selection probability.

In our optimization approach, each chromosome is composed of four inter-linked parts: the visiting order of points, the manipulator IKM configurations at each point and its relative placements and orientations. The near optimal solutions are selected based on the objective function calculated based on (4).

3.1. *Encoding process*

A GA based method requires an adequate representation to encode possible solutions of the optimization problem. This encoding can be implemented by either natural or binary alphabets. In our case, order of task-points visit part of the chromosome must be encoded by integer values, since the random selection process of the mutation may repeat on the same alphabets which creates serious conflict phenomena in the case of binary encoding. Moreover, in order to avoid the redundancy of the visited point part that could be caused by the crossover operation, we developed a specific algorithm to correct bits that lead to same genes. For IKM configuration of the manipulator and its relative placements and orientations, a binary encoding scheme has been implemented. For the sake of brevity, a 2–*DOFs* and a *n*-DOFs planar manipulators are exemplified below.

3.1.1. *2-DOFs planar manipulator*

Consider again the manipulator shown in Fig. 3 performing a task in 2*D* operational space. Suppose that the manipulator's base is placed at position corresponding to placement number 5 with orientation number 2. The manipulator can visit *N* task-points using two possible IKM configurations at each point. Fig. 8 depicts the four parts of the chromosome of this problem, where the chromosome is constructed by *N*, $N \times 2$, 2 and 2 genes (for brevity, hereafter we refer to gen using integers but not binary). In this

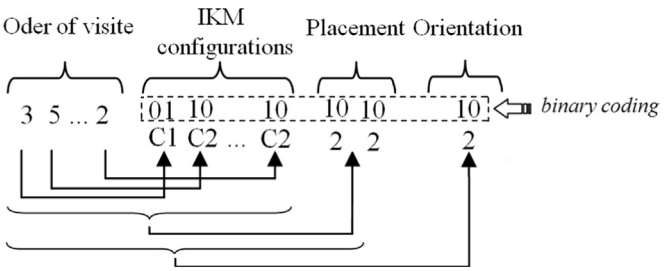

Fig. 8. Chromosome of 2-*DOFs* manipulator visiting *N* points in 2*D*.

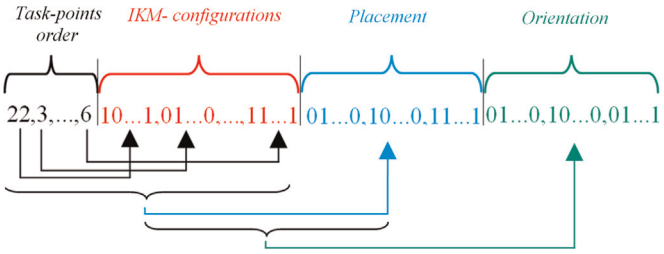

Fig. 9. Chromosome of *n-DOFs* manipulator visiting *N* points.

example, both the placement coordinates *X* and *Y* are equal to 2 which is "10" in binary encoding. The same for the IKM configurations (i.e., 1st and 2nd configurations are encoded by "01" and "10", respectively). The manipulator travels from point 3 to 5 using 1st (*C*1) and 2nd (*C*2) configurations at each of these points respectively. It finally finishes visiting the last point with the second configuration (*C*2).

3.1.2. *n-DOFs manipulator*

Fig. 9 shows the chromosome form of the problem discussed in Section 2. As shown, the manipulator starts performing the task by visiting the point number 22 using the configuration coded with the binary digits "10, …, 1". It then visits the point 3 using the configuration "01, …, 0" and finally finishes the task by visiting the point number 6 with configuration "11…1". The relative placement of the manipulator w.r.t. task-points is on nodes "01, …, 0", "10, …, 0" and "11, …, 1" corresponding to *X*, *Y* and *Z* axes respectively while the relative orientation of the manipulator configuration is situated on the nodes "01, …, 0", "10, …, 0" and "01, …, 1" around $X(\theta)$, $Y(\psi)$, and $Z(\phi)$ axes respectively.

3.2. *Optimization process*

In our approach, a population of 50 chromosomes is generated randomly to be reproduced over 800 generation. The evaluation process is applied according to the objective function described earlier, which is based on the cycle time that each solution can produce; where solutions with minimum time (high fitness value) are selected to be reproduced. The selection constraint is a rank based selection where an elitist fraction of 10% is preserved unchanged (no mutation or crossover) from the current to the next generation, to guarantee that the best solutions will not be lost. The remaining chromosomes of the population are reproduced by applying a mutation and crossover probability of 10% and 96%, respectively. The applied crossover is the uniform crossover, since it performs well compared to the two-point crossover, because this last leads to generate non-homogeneous chromosomes in our case.

4. Simulation setup

4.1. *3D robotized site modeling and task definition*

To evaluate the proposed approach, a real world industrial example of an automation plant with a 6-DOFs industrial manipulator (Stäubli RX-130 XL) has been selected. The task of spot-welding has been chosen because of its obvious importance in many industrial plants, especially in assembling car bodies. As shown in Fig. 10, it is necessary to develop the whole 3*D* model of the robotized site including a manipulator and its workspace, working environment, desired task-points and pieces to be assembled by the manipulator (car-body). Fig. 11 represents a flowchart that explain the followed stages to develop the simulation.

The coordinates of task-points are distributed on whole body of the Car. The end-effector (Fig. 10(b) left) position and orientation corresponding to task-points are developed using CAD-learning technique illustrated in [30,17]. The overall size of the occupied task zone is 1695.75 mm, 286.18 mm and 479.98 mm in *X*, *Y* and *Z* axes, respectively. After defining the 3D model, placement and orientation zones are calculated as discussed in [24]. The resulting variables corresponding to placement and orientation zones are 290.46 mm, 290.50 mm, 302.12 mm, 119.89°, 223.94° and 246.59° in *X*, *Y*, *Z*, θ, ψ and ϕ, respectively.

4.2. *Case study*

To assess the effectiveness of the proposed approach, cycle time of the task in four different stages (see Table 2) has been evaluated for quantifying influence of each part. While a fifth stage gives an overview of all the fourth stages. Consider the nomenclature as T: task-points order, C: IKM configurations, P: placement, O: orientation. The optimization problem is analyzed with different combinations (one by one, two by two, three by three and finally combining all parts together). Possible combinations of the chromosome's parts are listed in Table 2. The simulations are aimed to optimize cycle time required by the manipulator to perform the spot-welding at all task-points and finally return back to the initial configuration. For each combination of chromosome's parts, 10 trials have been conducted during evaluation process.

5. Results and discussion

The proposed optimization approach is examined by computing time for task accomplishment according to the plan detailed in Section 4. The results of first three stages are compared with

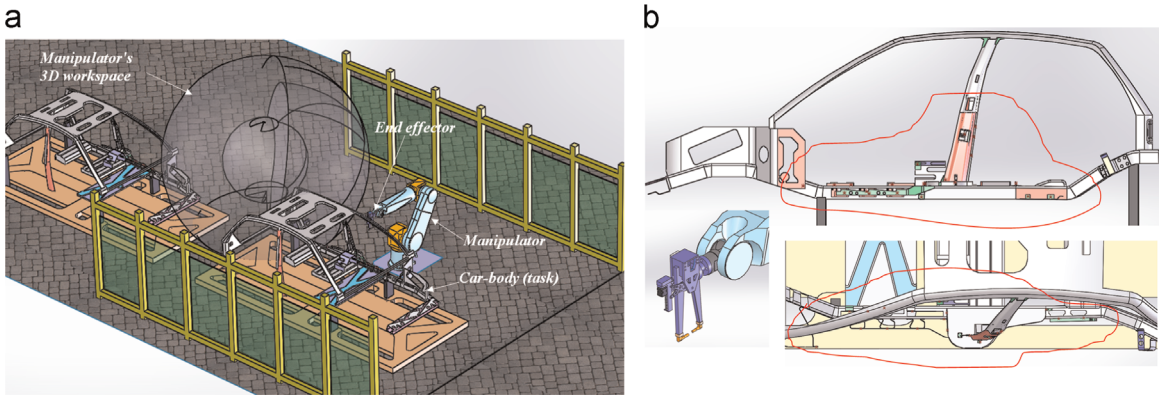

Fig. 10. 3*D* model of the robotized site developed in SolidWorks®. (a) Model of task, manipulator and working environment. (b) Various views of task-points and end-effector.

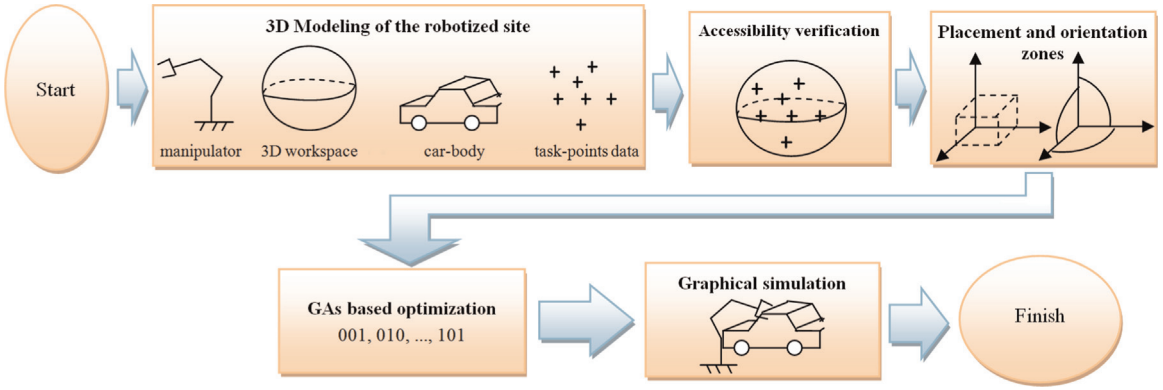

Fig. 11. A general schema of the case of study.

Table 2
Possible combinations of chromosome's parts.

First stage	T	C	P	O		
Second stage	T–C	T–P	T–O	C–P	C–O	P–O
Third stage	T–C–P	T–C–O	T–P–O	C–P–O		
Fourth stage	T–C–P–O					

analysis of variance (ANOVA) method for accurate analysis, while the forth is only one combination, and thus is not possible applying this technique in this context.

5.1. First stage

Starting with a random population, the first evaluation is calculation of cycle time according to individual contribution of each chromosome parts. Fig. 12 shows comparison results of the first stage. The difference in the lean value (of 10 trials) for the four parts can be clearly seen. T part gives the value of 39.34 s while C results in a better value of 30.97 s (best time in this comparison). P and O exhibit values of mean times very close to each other (45.22 s and 50.27 s, respectively). The *p-value* or the significance of the difference on the four parts is very small ($3.24e-21$) which indicates a considerable difference among these parts. This can also be simply observed from the maximum difference of cycle time among these combinations (19.3 s). The variance of C part looks smaller because during most of the trials, the optimization procedure has already resulted in the minimum value of cycle time. So, the best solution for a given combination cannot be improved further. This points that C part is very dominating as compared with others. However, the variance in values of cycle time corresponding to parts T, P and O is 6.02 s, 6.43 s and 6.54 s, respectively, which says that these parts can improve given solutions by resulting to more generations. The cycle time given by each part is strongly related to the fixed values of other parts.

5.2. Second stage

The second stage deals with cycle time of the task based on combinations of different parts taken two at a time. The number of chromosome becomes larger (2_*N genes) compared to the previous stage. Considering the first combination i.e., T and C, fixed values are assigned to P and O parts. Consecutively, we combined other parts i.e., T with P and then with O, by fixing C and P, respectively. Adopting same procedure for C, P and O hereafter, Fig. 13 shows the comparison of six resultant combinations. The *p-value* of $5.29e-023$ clearly indicates significance of the difference among these combinations. It is observed from the results that the combination of C with P and C with O gives better performance with the corresponding values of cycle time of 27.57 s and 27.61 s, respectively. This gives a hint that C–P performs relatively better. Also, larger variance in C–P (7.86 s) compared to that in C–O indicates diversity of the solutions found by the later combination. Some of these solutions achieve better performance than the best value given by C–O, as well all other combinations, indicating the possibility of finding better solutions. The variance of C–O is smaller which indicates that this combination converges to a near optimal solution with few exceptions e.g., see the outlier of 24.90 s

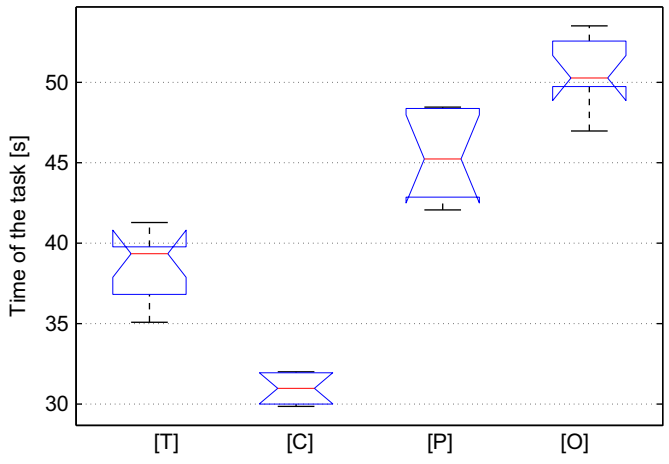

Fig. 12. ANOVA based comparison among the four parts used in optimization procedure individually.

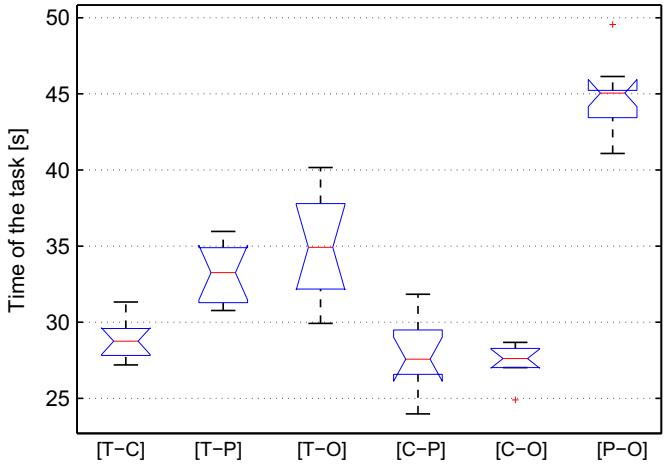

Fig. 13. ANOVA based evaluation of the task time by combining parts in a group of two.

in Fig. 13 which is less than mean value of C–O. In overall, C part seems to be more dominating by giving the third best time of 28.75 s in this series when is combined with T, i.e., T–C combination. The T part combined with P gives better results in comparison with the same part combined with O; values of the mean cycle times for T–P and T–O combinations are 32.26 s and 34.91 s, respectively.

Regarding the possibility to find better solutions, it seems that T–O performs better by offering an interval of solutions varying from 40.16 s to 29.92 s (with variance 10.24 s) over 10 trials. This draws attention toward huge space of solutions by considering O and the complexity of convergence i.e., it does not converge to the best possible solution at all trials. Comparing this to T–P, the solutions obtained in T–O are very diverse from each other thus allowing the algorithm to find best solution by playing with GAs parameter as reported in [17].

Finally, the combination of P and O seems to give bad performance due to the reduced space of solutions constrained by fixation of T and C parts. However, by considering the variance value of 5.06 s (the 3rd largest in this stage), the algorithm can offer best cycle time than the given mean value of the 10 trials. Generally, P and O enhance the performances of T and C during optimization process, respectively. For instance, in comparison with the first stage, T reduces the cycle time from 39.34 s to 32.26 s and to 34.91 s when combined with P and O parts, respectively. Similarly C improves the cycle time from 30.97 s to 27.57 s and to 27.61 s when combined with P and O parts, respectively. These improvements do not consider essential parts such as T in case of combinations C–P and C–O and C for combinations of T–P and T–O.

5.3. Third stage

This stage combines parts in a group of three to quantify the effect of different combinations on cycle time. So, the level of complexity related to the search space is increased because of considering more parts. The size of chromosomes becomes much bigger as compared to the previous stages. Also, the optimization process in this stage performs well for finding a near optimal solution compared with the previous stages as illustrated in Fig. 14. The best value of mean cycle time is offered by the combination of T with C and P parts, where, the cycle time is reduced to 22.37 s. While the second and third best times in this stage are given by the combinations of T with C and O (25.89 s) and C with P and O parts (26.58 s), respectively. The worst time of 31.18 s is given by the combination of T with P and O parts. It is worth to notice the performance improvement of the algorithm in third stage as compared to previous two stages. In detail, the combination T–C improves when combined with P by reducing cycle time from 28.75 s to 22.37 s. Similarly, O part when added in combinations such as T–P and C–P improves cycle time. This may give a hit that adding P to some combinations give better solutions than O. When O part is added to T–P and C–P, the mean time seems to be slightly improved. However, considering the variance of T–C–O (3.77 s), we definitely expect that the algorithm can lead towards better solutions especially by tuning the control parameters. Based on observation, we can see that O part improves the results in many other trials, such as T–P–O.

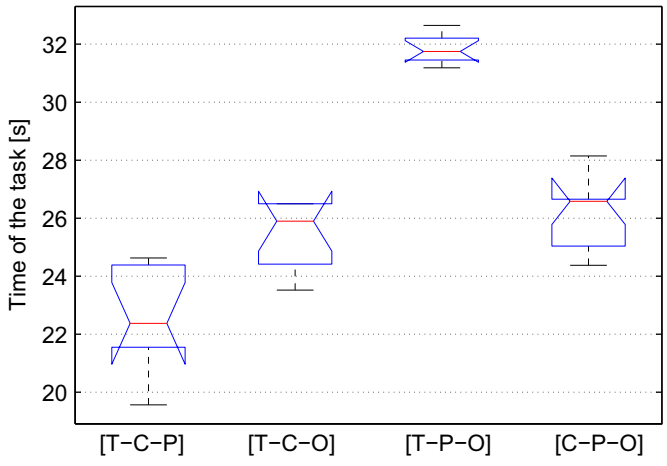

Fig. 14. ANOVA based evaluation of the task time by combining parts in a group of three.

5.4. Fourth stage

In this stage, all the parts are combined together in order to allow them to beneficiate from one another to enhance the performances of the optimization process in searching near optimum cycle time. Moreover, we compare the outcomes of activating the parts serially with that of simultaneous activation. In both cases, we show the best and the worst times at each generation, at one of the best trials. Initially, the parts are activated one after the other, where each part starts from the population found by the previous part. Thus each parts is activated through 200 generations. In this way we allow all parts to participate in the optimization process but not simultaneously (i.e., successively). Fig. 15(a) shows corresponding optimization results. The cycle time is reduced to 29.56 s. This result is better than the one given by arrangement of the first stage, where the best value reaches up to 30.97 s. This looks normal since solutions given by first stage is improved successively by activating the other parameters. However, this results is no longer better than the results achieved by the second and the third stages, where up to two and tree parameters are considered simultaneously. This illustrates the significance of launching all parts together, since activating only two parts simultaneously achieves the same results as that of serial activation.

In case we launch all parts together simultaneously, the chromosome size becomes even larger. This successfully increases the chance of finding relatively better results. However, it may take more computational time in terms of algorithm convergence. Fig. 15(b) shows corresponding results, where the best cycle time reaches the value of 18.46 s, while the worst time value is 86.02 s. This is an impressive result when compared with previous stages that deal parts individually or in a group of two or three. Also it is clear that the near optimal time is a result of contribution of all parts together, where each one helps the others to find the good sequence by increasing the search space and performance of the search process. The cycle time in this stage has been reduced from 42.04 s (found in first generation) to 18.46 s (found in last generation) with a difference of 23.58 s. Considering the number of task-points of 22 in the present work, the result obtained is very promising when compared with the results reported in [17], where the cycle time was 7.66 s with the number of task-points as 12 and without consideration of orientation. Defining a ratio of the reduced time per task-point, consequently, this ratio for the proposed algorithm is 1.0718 s/*point*, while in case of research reported in [17], it is 0.6383 s/*point*. Moreover, this explicitly indicates that the proposed algorithm performs very well, even considering relatively high number of task+points.

5.5. Fifth stage

An overall view of all combinations of parts is illustrated in Fig. 16, where the mean of cycle time offered by each stage is depicted (Fig. 16(a)). This may allow us to quantify the combinations and classify them based on which set of parts one has more influence on the optimization process. Observing the mean of the

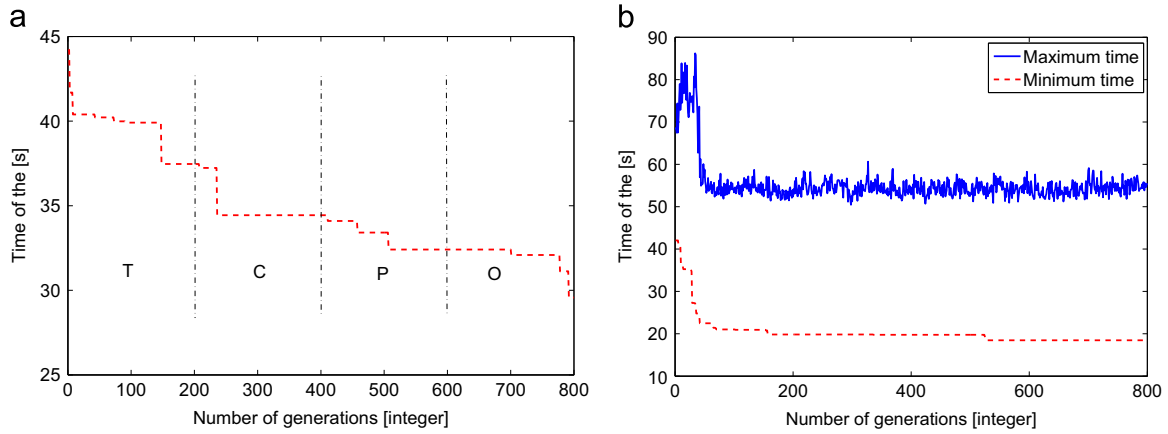

Fig. 15. Results of maximum and minimum cycle times within an optimization procedure by activating parts: (a) one by one successively and (b) simultaneously.

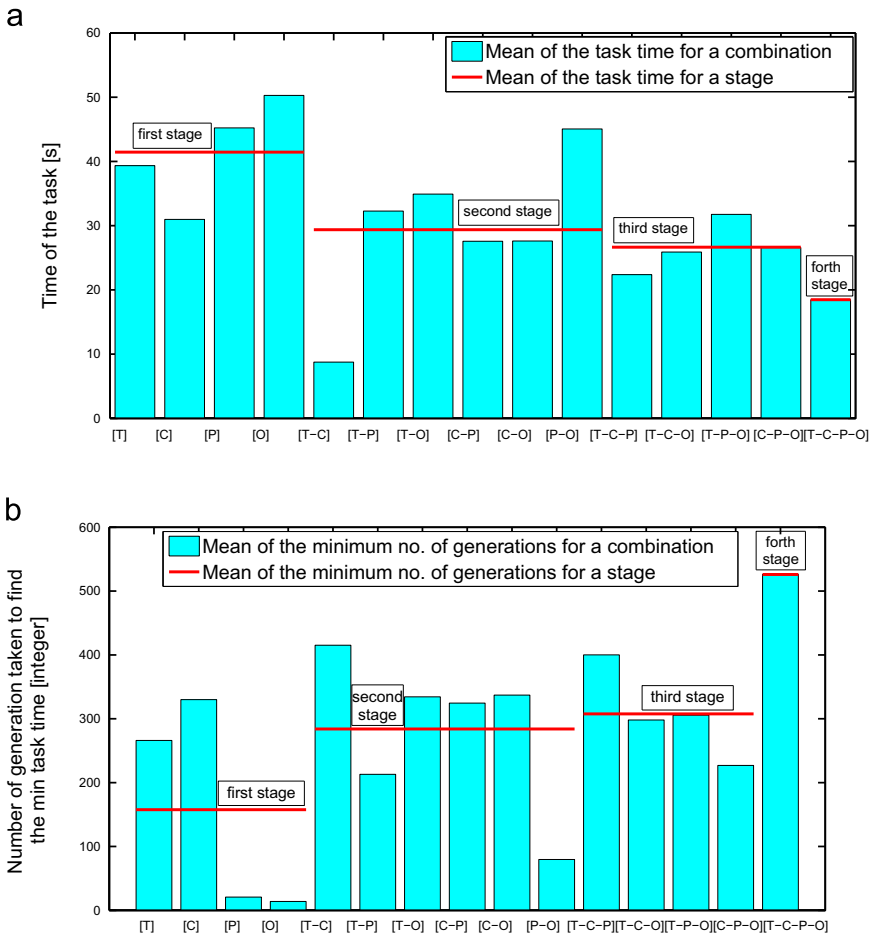

Fig. 16. Comparison of all stages and all combinations of parts w.r.t.: (a) mean of the near optimal task execution time. (b) Mean of the minimum number of generations to find the near optimal solution.

best times given by each combination and each stage, it is clear that combining different parts brings significant benefits in the optimization algorithm by improving each chromosome of the population to achieve the search and fine a near optimal value of cycle time. The best cycle time found in *fourth stage* is very less as compared to the previous stages. Contrary to cycle time, the computational time, which is a function of the minimum number of generations to find the best solution, is increased. This is because of increase in the search space, and even with higher number of generations (at the order of 500) the algorithm is still capable of finding alive solutions. Fig. 16(b) illustrates the number of generations needed to find a near optimal solution.

5.6. Graphical simulation

The robotic task was graphically simulated using our SimulationRobot platform that was developed under SolidWorks® API (Fig. 17(a) bottom). Fig. 17 shows a sequence of achievement of the manipulator while performing the spot-welding of an industrial car-body assembly task. For brevity we limited this figure on few points.

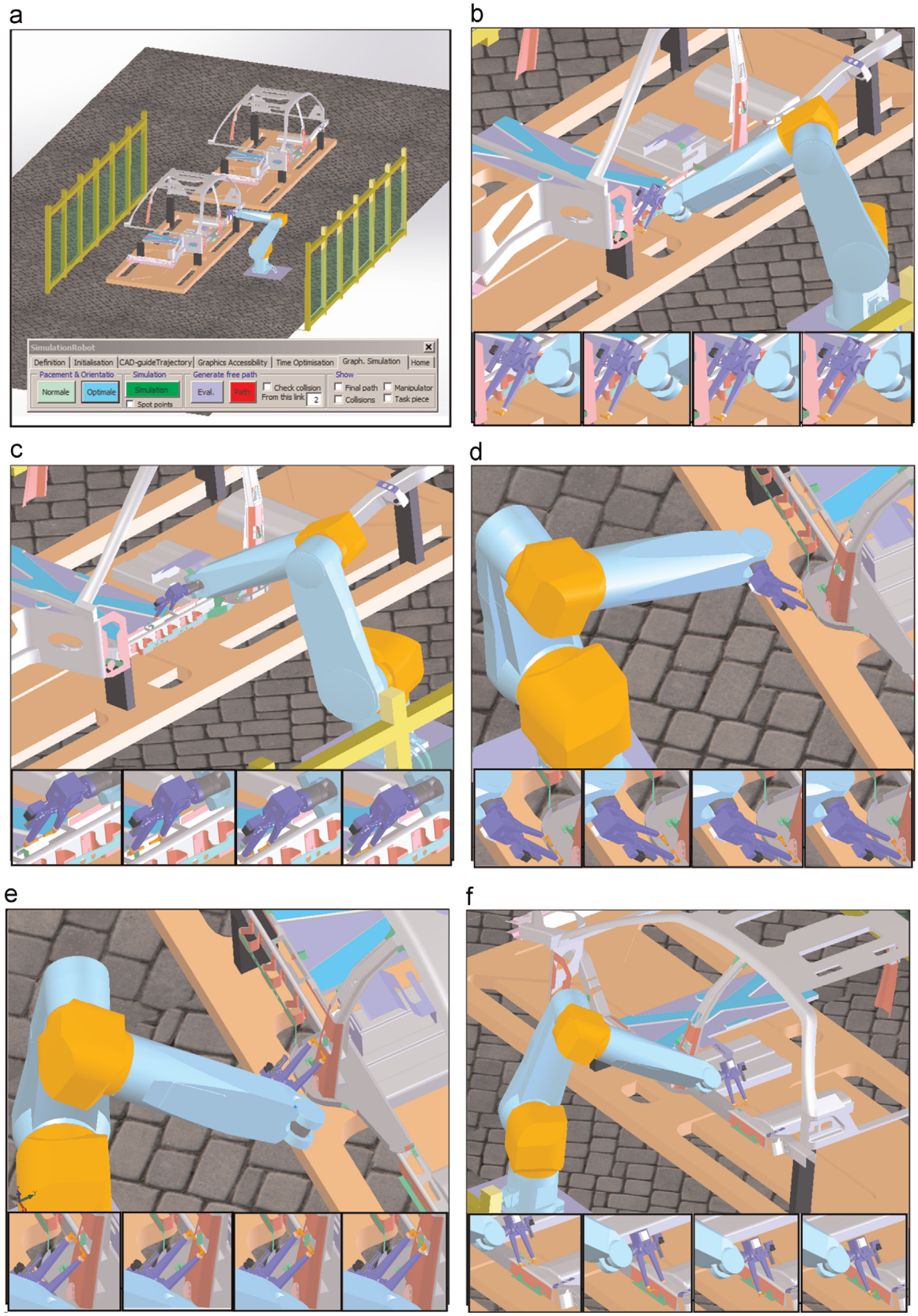

Fig. 17. Some snapshots of the graphical simulation of a spot-welding task: (a) initial configuration, (b) first point, (c) fifth points, (d) 11th point, (e) 12th point, (f) last point, snapshots of the welding at each point are from left to right.

6. Conclusion

This paper presents a novel approach of time scheduling and optimization based on GAs with a focus on industrial robotized tasks. The proposed strategy takes into consideration multitude parameters that have a direct effect on cycle time of the task such

as task-points order, IKM configurations used at each task-point, manipulator s placement and its orientation w.r.t. the task. With the objective to have precise results, a direct comparison among all different combinations of these parts has been performed using ANOVA to quantify the quality of each part in reducing cycle time. Results of these comparisons show that w.r.t. the influence on cycle time, the parts can be arranged in descending order as IKM configurations, task-point order, placement and orientation. The last two parts enhance quality of the solution by increasing the search space. The cycle time has been reduced considerably from 42.04 s to 18.46 s. The tremendous increase in the demands of cars due to rise in the population size world-wide has necessitated production of cars at an incredible rate. As reported in [31], the rate of car manufacturing in an automotive plant can go up to 58 cars/h (i.e. 1512 cars in 24 h). Indeed the reduction of the cycle time, given above, is very significant. e.g., considering the automobile plant cited above. Halving cycle time of the robotics chain significantly helps to double the production per day. This benefit is not negligible given that the alternative cost of production corresponding to second chain (infrastructure, personals, maintenance, etc.) is dramatically higher. Moreover, in spite of wide search space, the computational time has been reduced to the order of minutes instead of months and years needed to scan all possible solutions.

References

[1] G.S. Tewolde, W. Sheng, Robot path integration in manufacturing processes: genetic algorithm versus ant colony optimization, IEEE Trans. Syst. Man Cybern. 38 (2) (2002) 278–287.

[2] M. Dissanayake, J. Gal, Workstation planning for redundant manipulators, Int. J. Prod. Res. 32 (5) (1994) 1105–1118.

[3] L. Abdel-Malek, Z. Li, The application of inverse kinematics in the optimum sequencing of robot tasks, Int. J. Prod. Res. 28 (1) (1990) 75–90.

[4] Y. Edan, T. Flash, U. Peiper, I. Schmulevich, Y. Sarig, Near minimum-time task planning for fruit-picking robots, IEEE Trans. Robot. Autom. 7 (1) (1991).

[5] P.Th. Zacharia, N.A. Aspragathos, Optimal robot task scheduling based on genetic algorithms, Robot. Comput.-Integr. Manuf. J. 21 (1) (2005) 67–79.

[6] H. Chen, W. Sheng, N. Xi, M. Song, Yifan Chen, CAD-based automated robot trajectory planning for spray painting of free-form surfaces, Int. J. Ind. Robot 29 (5) (2002) 426–433.

[7] J.E. Bobrow, S. Dubowsky, J.S. Gibson, Time optimal control of robotic manipulators along specified paths, Int. J. Robot. Res. 4 (3) (1985) 3–17.

[8] K. Shin, N. Mckay, Selection of near minimum time geometric paths for robotic manipulators, IEEE Trans. Autom. Control 31 (6) (1986) 501–511.

[9] K. Baizid, R. Chellali, T. Bentaleb, A. Yousnaj, A. Meddahi, Virtual reality based tool for optimal robot placement in robotized site based on CADs application programming interface, in: Proceedings of Virtual Reality International Conference (VRIC 2010), Laval, France, 7–9 April 2010.

[10] Pedro Neto, J.N. Pires, A.P. Moreira, CAD-based off-line robot programming, in: 2010 IEEE Conference on Robotics Automation and Mechatronics (RAM), 28–30 June 2010, pp. 516, 521.

[11] Kin-Huat Low, Industrial Robotics Programming, Simulation and Applications.

[12] E. Kahn, B. Roth, The near-minimum-time control of open-loop articulated kinematic chains, ASME J. Dyn. Syst. Meas. Control 39 (3) (1971) 164–172.

[13] K.G. Shin, N.D. McKay, Minimum-time control of robotic manipulators with geometric path constraints, IEEE Trans. Autom. Control 30 (6) (1985) 531–541.

[14] S. Dubowsky, T. Blubaugh, Planning time-optimal robotic manipulator motions and work places for point-to-point tasks, in: IEEE Conference on Decision and Control, Ft. Lauderdale, FL, 1985.

[15] J.F. Petiot, P. Chedmail, J.Y. Hascot, Contribution to the scheduling of trajectories in robotics, Robot. Comput.-Integr. Manuf. 14 (3) (1998) 237–251.

[16] E.K. Xidias, P.T. Zacharia, N.A. Aspragathos, Time-optimal task scheduling for articulated manipulators in environments cluttered with obstacles, Robotica 28 (03) (2010) 427–440.

[17] K. Baizid, R. Chellali, A. Yousnadj, A. Meddahi, B. Toufik, Genetic algorithms based method for time optimization in robotized site, in: IEEE/RSJ International Conference on Intelligent Robots and System (IROS), 18–22 October 2010, Taiwan.

[18] E. Lawer, J. Lenstra, A. Rinnooy Kan, D. Shmoys, The Travelling Salesman Problem, Wiley, Chichester, UK, 1985.

[19] H.S. Hwang, An improved model for vehicle routing problem with time-constraint based on genetic algorithm, Comput. Ind. Eng. 42 (April (2–4)) (2002) 361–369.

[20] L. Qu, R. Sun, A synergetic approach to genetic algorithms for solving traveling salesman problem, Inf. Sci. 117 (August (3–4)) (1999) 267–283.

[21] F. Liu, G. Zeng, Expert Syst. Appl. 36 (April (3–2)) (2009) 6995–7001.

[22] Paraskevi Th. Zacharia, Elias K. Xidias, Nikos A. Aspragathos. Task scheduling and motion planning for an industrial manipulator, Robot. Comput.-Integr. Manuf. 29 (December (6)) (2013) 449–462.

[23] S. Alatartsev, V. Mersheeva, M. Augustine, F. Ortmeier, On optimizing a sequence of robotic tasks, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 3–7 November 2013, pp. 217, 223.

[24] K. Baizid, R. Chellali, A. yousnadj, A. Meddahi, B. Toufik, J. Iqbal, Modelling of robotized site and simulation of robots optimum placement and orientation zone, in: 21th Modelling and Simulation, Canada, 15–17 July, 2010.

[25] Wisama Khalil, Etienne Dombre. Modlisation identification et commande des robot HERMES Science Publications 75004, Paris.

[26] L. Tsai, A. Morgan, Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods, in: ASME Mechanics Conference, Boston, 7–10 October 1984.

[27] H.-Y. Lee, C. Woernle, M. Hiller, A complete solution for the inverse kinematic problem of the general 6R robot manipulator, ASME Trans. J. Mech. Design 113 (1991) 481–486.

[28] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press, Cambridge, MA, USA, 1992.

[29] Z. Michalewitz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed., Springer, Berlin, 1996.

[30] A. Meddahi, K. Baizid, A. Yousnadj, J. Iqbal, API based graphical simulation of robotized sites, in: IASTED International Conference on Robotics and Applications, Cambridge, USA, 2009, pp. 485–492.

[31] Lesechos, ⟨http://www.lesechos.fr/07/05/2013/LesEchos/21432-068-ECH_une-production-optimisee-au-maximum.htm⟩ (accessed 06.12.14).