International Journal of Computational Intelligence and Applications
Vol. X, No. X (200X) 1-19

TIME-OPTIMAL PATH PLANNING AND CONTROL USING NEURAL NETWORKS AND A GENETIC ALGORITHM

NACHOL CHAIYARATANA*

Research and Development Center for Intelligent Systems,
King Mongkut's Institute of Technology North Bangkok, Thailand

ALI M. S. ZALZALA†

Department of Computing and Electrical Engineering, Heriot-Watt University, UK

This paper presents the use of neural networks and a genetic algorithm in time-optimal control of a closed-loop 3-dof robotic system. Extended Kohonen networks which contain an additional lattice of output neurons are used in conjunction with PID controllers in position control to minimise command tracking errors. The extended Kohonen networks are trained using reinforcement learning where the overall learning algorithm is derived from a self-organising feature-mapping algorithm and a delta learning rule. The results indicate that the extended Kohonen network controller is more efficient than other techniques reported in early literature when the robot is operated under normal conditions. Subsequently, a multi-objective genetic algorithm (MOGA) is used to solve an optimisation problem related to time-optimal control. This problem involves the selection of actuator torque limits and an end-effector path subject to time-optimality and tracking error constraints. Two chromosome coding schemes are explored in the investigation: Gray and integer-based coding schemes. The results suggest that the integer-based chromosome is more suitable at representing the decision variables. As a result of using both neural networks and a genetic algorithm in this application, an idea of a hybridisation between a neural network and a genetic algorithm at the task level for use in a control system is also effectively demonstrated.

Keywords: Kohonen Network, Multi-Objective Genetic Algorithm, Time-Optimal Control, Robotics

1. Introduction

Time-optimal control has been one of the major research interests in robotics during the past decade. Time-optimality can lead to an overall improvement in the level of productivity from a manufacturing viewpoint and an increase in the effectiveness of a task execution from an operational viewpoint. One particular aspect of research is the theory and application of time-optimal control of a robot arm along a pre-defined path. An algorithm that can lead to time-optimality of this kind was firstly developed

* E-mail: nchl@kmitnb.ac.th
† E-mail: a.zalzala@hw.ac.uk

by Bobrow *et al*.[1] Over the years, this algorithm has undergone a number of refinements and one of the latest modifications has been described in Shiller and Lu.[2] In summary, a time-optimal motion of a robot arm along a pre-defined path is achieved when the motion is executed with either the maximum possible acceleration or deceleration along the path. This can be done when one of the actuators on the robot arm is always saturated and the other actuators adjust their torque values so that their torque limits are not violated.[3]

Although this time-optimal control algorithm has been proven to be a useful algorithm in a number of robotic applications, the majority of the demonstrations have only been done in the open-loop control mode. This can hardly be the case for a practical use of motion control in a real-time implementation where closed-loop control would be a more common practice. Shiller *et al*.[4] have pointed out that the actuator dynamics and the delays caused by an on-line feedback controller would lead to a reduction in the efficiency of the algorithm when closed-loop control is used. Three possible methods have been used to solve this problem. The first method is based on a modification of the original time-optimal control problem into a time-energy optimal control problem which can be regarded as a lagrangian constraint optimisation problem and can only be solved numerically.[5] A drawback of this method is that the modification also leads to an increase in the resulting trajectory time. The second method is based on the use of a simplified friction model to compensate for the actuator dynamics and the implementation of a trajectory pre-shaping to account for the dynamics of the controller.[4] Finally, the third method covers the use of a neural network which is trained using feedback error learning[6] as an additional controller in the control loop. The primary function of this neural network is to compensate for modelling errors and delays caused by the main controller in the system. It has also been demonstrated that the compensation performance of the neural network controller is higher than that of the trajectory pre-shaper.[7]

The work initiated by Chaiyaratana and Zalzala[7] will be continued in this paper where the investigation will cover the use of time-optimal control in a closed-loop 3-dof robotic system. Similar to the earlier work, the investigation will be carried out in a similar way to that described in Shiller *et al*.[4] except that the actuator dynamics are not considered. The neural network controllers will be used in conjunction with the standard controllers, which leads to the redundancy of the use of trajectory pre-shaper. In contrast to the earlier work where the feedback error learning is used, in this paper the neural network controllers will be trained using reinforcement learning. In addition to the continuation on the study of neural network capability in the compensation task, a further multi-objective optimisation problem associated with the use of time-optimal control is also considered. This problem is inspired by an observation that many tasks in manufacturing systems can be accomplished using lesser processing time provided that the trade-off in the product quality is acceptable. For instance, in tasks like welding and edge-deburring, the time that the robot end-effector required to track the pre-programmed path can be reduced if the allowable tracking error bound is increased. This is the problem that will be addressed in this paper. Note that this is an extension to the multi-objective problem presented in Chaiyaratana and Zalzala.[7] The optimisation problem interested involves the selection of torque limit combination and the path planning process where the search objectives

are expressed in terms of the position tracking error and trajectory time. An approach on multi-objective optimisation using a genetic algorithm, namely a multi-objective genetic algorithm (MOGA)[8] will be used to solve the mentioned problem. Since the neural network and genetic algorithm are used in the different part of the control application, in essence this indicates a task hybridisation between a neural network and a genetic algorithm.

This paper is presented as follows. The time-optimal control algorithm as described by Shiller and Lu[2] is briefly explained in section 2. In addition, the trajectory pre-shaping scheme is also explained in this section. In section 3, the overview of the time-optimal control problem is discussed. In section 4, the control structure of the robotic system and the neural network contribution is given. The improvement in the system performance gained by using neural network controllers and the comparison with the previous results reported in Chaiyaratana and Zalzala[7] is illustrated in section 5. The multi-objective optimisation problem associated with time-optimal control and the MOGA are explained in section 6. The optimisation results and the related discussions are given in section 7. Finally, the conclusions are drawn in section 8.

2. Time-Optimal Control Algorithm and Trajectory Pre-Shaping Scheme

In summary, time-optimal control algorithm as described by Shiller and Lu[2] can be used to generate the time-optimal profiles of the reference joint position and the open-loop control torque signal provided that the physical properties of the robot arm are known and a pre-defined path of the robot arm in the workspace is available. In particular, the torque limits on the actuators within the robot are the key factors which have a major influence on the trajectory time obtained from the algorithm. As stated earlier, the time-optimal motion is achieved when one of the actuators on the robot arm is always saturated and the torque values of other actuators are within the bounds of the corresponding limits. This means that with the large values of the torque limits, the obtained trajectory time will be short. On the other hand, with the smaller values of the torque limits, the obtained trajectory time will be relatively larger. A schematic diagram describing input and output of the time-optimal control algorithm is given in Fig. 1. In Fig. 1, the time-optimal control algorithm takes the robot physical properties and the information regarding the pre-defined robot's path as inputs. The outputs from the algorithm are the reference joint position and the open-loop torque profiles.

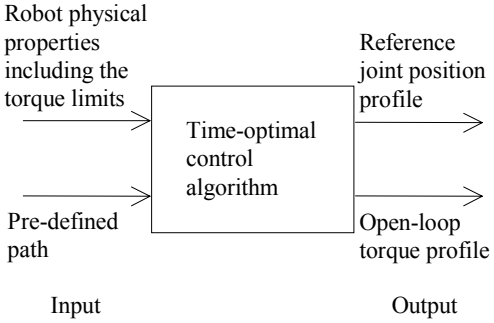


Fig. 1. Schematic diagram of the time-optimal control algorithm.

Nonetheless, the time-optimal control algorithm will produce a result based on the open-loop dynamics of the system. This means that a certain number of problems will arise when using the reference joint position profile obtained from the algorithm as input to the closed-loop system.[4,5] In order to solve the problem, Shiller *et al*.[4] have introduced a method known as trajectory pre-shaping which involves a modification of the reference joint position profile according to the dynamics of the closed-loop system. This modification involves adding the open-loop reference joint position profile with a factor given by the open-loop torque profile which has been transformed by the inverse model of the controller in the closed-loop system. This modified or "pre-shaped" reference joint position profile is then used as input to the position feedback system in the usual way. Although some good results obtained by using trajectory pre-shaping have been reported in early literature, it will be demonstrated in sections 4 and 5 that the use of neural networks to compensate for dynamics of the controllers and modelling errors helps to remove the need for trajectory pre-shaping.

3. Overview of the Time-Optimal Control Problem

The simulations which are used to demonstrate the functions of a neural network and a genetic algorithm in the time-optimal control application involve the use of a 3-dof robot in a position control task. A schematic diagram of the 3-dof robot and its physical properties are given in Fig. 2. This position control task requires the robot to track a one-metre straight-line path; this is illustrated in Fig. 3. Referring to Fig. 3, point *A* (0.736, 0.226, 0.093) is the initial location of the robot end-effector and point *B* (0.0, 0.854, 0.354) is the final desired location of the robot end-effector on this path. The time-optimal control algorithm is then used to generate the trajectory time history, which is subsequently used as the input to the position control loop.

4. Control Structure and Neural Network Contribution

Firstly, consider the dynamic equation of motion for an *n*-dof robot which is given by

$$\mathbf{D}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{h}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{c}(\boldsymbol{\theta}) = \mathbf{u}(t) \tag{1}$$

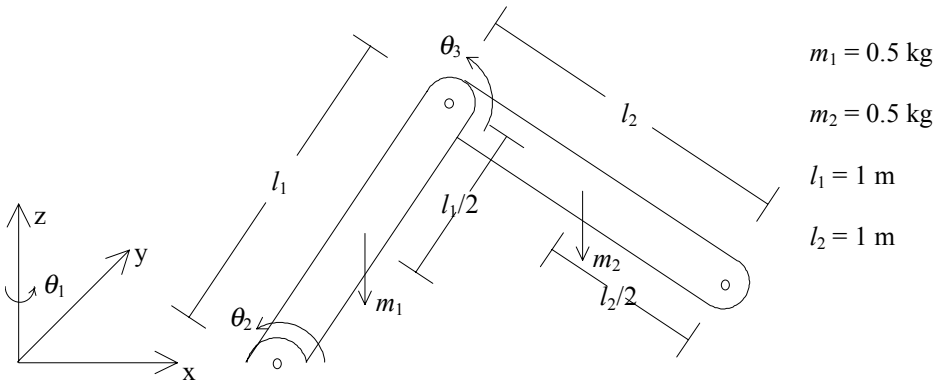


Fig. 2. Schematic diagram of the 3-dof robot.

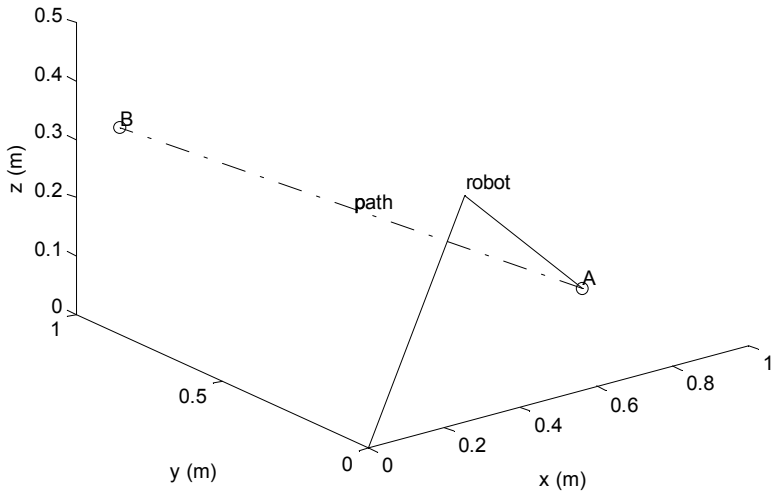


Fig. 3. Robot and the straight-line path.

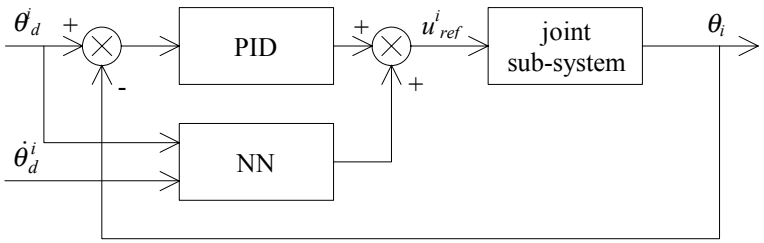


Fig. 4. Neural network and PID controllers in each joint control loop.

where $\mathbf{D}(\boldsymbol{\theta})$ is the $n \times n$ inertial acceleration-related matrix, $\mathbf{h}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is the $n \times 1$ centrifugal and Coriolis forces vector, $\mathbf{c}(\boldsymbol{\theta})$ is the $n \times 1$ gravity loading force vector, $\mathbf{u}(t)$ is the $n \times 1$ torque input vector, $\boldsymbol{\theta}(t)$ is the $n \times 1$ angular position vector, $\dot{\boldsymbol{\theta}}(t)$ is the $n \times 1$ angular velocity vector, $\ddot{\boldsymbol{\theta}}(t)$ is the $n \times 1$ angular acceleration vector and n is the degree of freedom of the robot model. Equation (1) indicates a non-linear relationship between the input torque and the joint angular parameters. The control strategy which is used in this study is the non-linear de-coupled feedback control. In this case, the control objective is to find a control signal $\mathbf{u}(t)$ such that the overall robotic system will be de-coupled into n linear second order systems. Freund[9] has suggested such a control signal which takes the form of

$$\mathbf{u}(t) = \mathbf{h}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{c}(\boldsymbol{\theta}) - \mathbf{D}(\boldsymbol{\theta}) \begin{bmatrix} \alpha_{11}\dot{\theta}_1(t) + \alpha_{01}\theta_1(t) - \lambda_1 u_{ref}^1(t) \\ \vdots \\ \alpha_{1n}\dot{\theta}_n(t) + \alpha_{0n}\theta_n(t) - \lambda_n u_{ref}^n(t) \end{bmatrix} \tag{2}$$

where α_{ij} and λ_i are arbitrary scalars. With the use of $\mathbf{u}(t)$ of this form, the overall dynamics of the system as described in Eq. (1) will transform into

$$\ddot{\theta}_i(t) + \alpha_{1i}\dot{\theta}_i(t) + \alpha_{0i}\theta_i(t) = \lambda_i u^i_{ref}(t),\ i = 1, 2, \ldots, n \qquad (3)$$

which indicates the de-coupled input-output relationship of the system. Using this form of de-coupling and non-linear compensation, each de-coupled joint sub-system can be controlled using a standard PID controller. In addition, a neural network can be used as an additional controller in each joint control loop where it will have a role of compensating for the dynamics of the primary controller and the possible modelling errors. This arrangement is illustrated in Fig. 4. However, with the control scheme as shown in Fig. 4, it is not possible to derive an exact desired neural network output training signal. Hence, an alternative training signal must therefore be acquired. One possible way for deriving an appropriate neural network output signal for use as an additional control signal is to use a reinforcement learning paradigm; this can be done as follows.

One procedure which can be used to accomplish reinforcement learning is a generate-and-test process. Basically, this process begins by generating a possible value of the neural network output. This newly generated neural network output is then combined with the control signal from the PID controller and subsequently applied to the model of joint sub-system where the predicted value of the command tracking error can be obtained. This command tracking error will be represented in the form of the reward value achieved by using the generated neural network output. In a similar manner, the unmodified value of the neural network output is also applied to the same model of the joint sub-system where another predicted value of the command tracking error and the associated reward value can also be obtained. If the change in the reward function – the difference between the two reward values – meets the criteria for adjusting the network connection weights, the network parameters will undergo an adaptation using an appropriate learning rule such as an error correction learning rule. After the adaptation, the network will send out the output signal which is being calculated using the updated settings of the network parameters where this output signal will be used as a part of the overall control signal for the actual robot. On the other hand, if the change in the reward function does not satisfy the criteria for the network adaptation, the network parameters will remain unchanged. The output from the network will then be calculated based on the unmodified settings of the network parameters. This process will continue until there are no changes in the network parameters. The schematic diagram of the reinforcement learning paradigm described above is illustrated in Fig. 5.

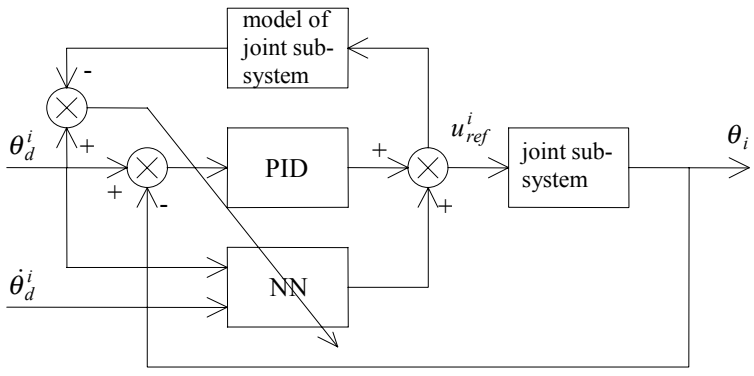


Fig. 5. Model-based reinforcement learning within the control loop of joint *i*.

In this study, Kohonen networks with an additional lattice of output neurons or the extended Kohonen networks[10] are used to assist PID controllers in the position control loop. A schematic diagram of the operation of the extended Kohonen network is illustrated in Fig. 6. In Fig. 6, the neurons in the additional lattice or the motor map are placed at the consecutive layer to the neurons in the original lattice or the state map. Notice that the number of neurons and the topological arrangement of neurons in the motor map are exactly the same as those in the state map. This means that once the state map is activated and the winning neuron in the state map is located, the output from this neuron will directly sensitise a neuron in the motor map which is located at the same corresponding site on the lattice. Once the neuron in the motor map is activated, it will send out the output signal which will subsequently be used as the control signal. Hence the only adjustable parameter within each neuron in the motor map will be in the form of a single connection weight.

As mentioned earlier, the model-based reinforcement learning is used to train the connection weights within the networks. The overall learning algorithm is derived from a self-organising feature-mapping (SOFM) algorithm and a delta learning rule. The SOFM algorithm is used to update the connection weights in the state map while the delta learning rule is used to adjust the connection weights in the motor map. Three neural network controllers, one for each joint sub-system, are trained and tested for use in position control of the 3-dof robot using a combination between the time-optimal position and velocity trajectories as both the training and testing samples. Note that this time-optimal trajectory is obtained for a robot task of tracking a straight-line path in Cartesian space shown in Fig. 3 with the torque limits on joints 1,

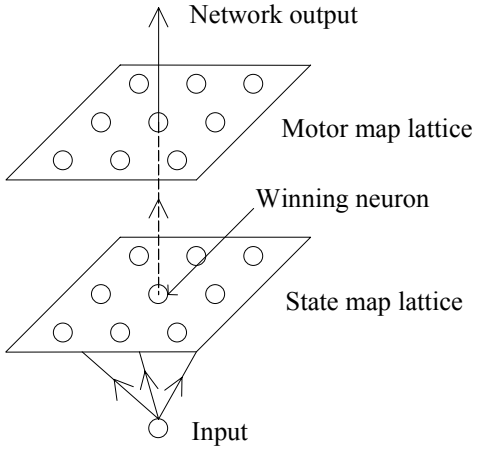


Fig. 6. Operation of the extended Kohonen network.

Table 1. Parameter settings for training neural networks.

Parameter	Value
Number of neurons in each network	98
Number of connection weights in each network	147
Number of input nodes in each network	2
Number of firing output nodes in each network	1
Number of training samples	30
Number of training epochs	400

2 and 3 of ±15, ±25 and ±5 Nm, respectively. The parameter settings for training neural networks are summarised in Table 1. The simulation results are displayed and discussed in the following section.

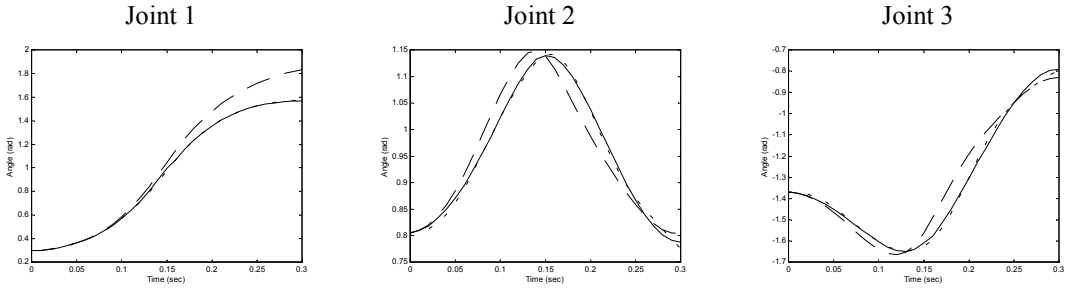
5. Results from Using Neural Network Controllers and Discussions

In this section, the simulation results from using the extended Kohonen network controllers will be discussed. In order to make the comparison, the results obtained using other techniques including the results achieved via the use of trajectory pre-shaping scheme and radial-basis function networks[7] will also be displayed alongside. Firstly, the simulation results for the case of PID controllers with trajectory pre-shaping and the case of PID and extended Kohonen network controllers are shown in Figs. 7, 8 and 9. In Figs. 7 and 8, the simulation results indicate that with the use of the extended Kohonen network controllers as the assistants to the PID controllers, a significant improvement in the control performance over that achievable by using trajectory pre-shaping mechanism can be observed. In Fig. 9, with the use of the extended Kohonen network controllers, the characteristics of the closed-loop torque profiles are similar to those of the open-loop control. This indicates that the time-optimality has been achieved within the torque constraints. Note that these trained extended Kohonen networks are used in the following parts including the following multi-objective optimisation problem without any further training.

Another advantage gained by using neural networks as assistants to PID controllers is the resistance to modelling errors which can occur during the robot operation. Many forms of error can be introduced to the robot system after the controllers have been designed. For example, a liquid spillage from the container attached to the last link of the robot arm can occur after an unexpected event, such as a collision. This kind of malfunction can lead to a loss of the overall mass in the last link of robot, which is a form of modelling error. This kind of modelling error is used in the following test cases which in turn are used to demonstrate the effectiveness of extended Kohonen network controllers in this kind of situation. In summary, in the following five test cases, a certain amount of mass in the last link, ranging from 10 % to 50 % of the overall mass, is lost during the operation. Note that the modelling error will only effect the mass of the last link and not the length of the last link. This makes the robot trajectory unaffected by this modelling error.

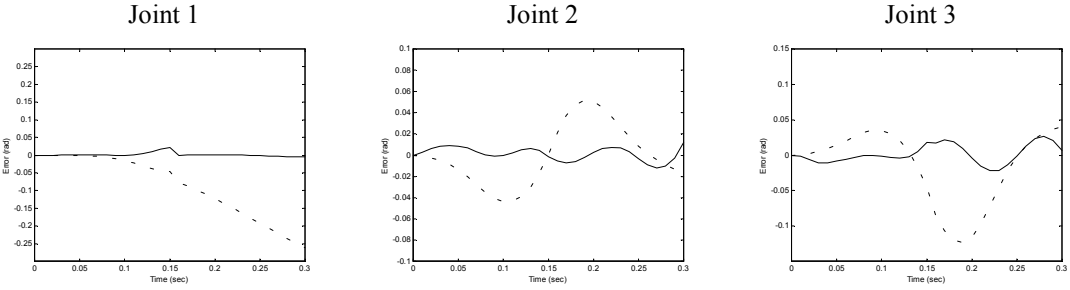
Since the modelling error occurs after the control structure has been determined, the robot physical model which is used in the time-optimal trajectory generation and feed-forward compensator for de-coupling the robot dynamics will remain unchanged. Also the same torque limits on the actuators will be used in the time-optimal trajectory generation. However, the generated reference position trajectory and open-loop torque profile will no longer be time-optimal. Nevertheless, it is sufficed to say that although time-optimality cannot be maintained after the occurrence of the fault, the robot operation can still be described as time sub-optimal. A summary of simulation results, expressed in terms of the sum of the mean absolute tracking errors from three joints over the whole trajectory, from all five test cases and the previous simulation with no mass loss is given in Table 2. Note that the simulation results from using trajectory pre-shaping and radial-basis function network controllers which are trained using feedback error learning[7] are also given for

comparison purposes. Again from Table 2, it is noticeable that in all test cases, the use of the neural networks as assistants to PID controllers has proven to be a more effective method in reducing tracking errors than the use of trajectory pre-shaping scheme. This indicates that neural network controllers are more suitable to the time-optimal control application both in the normal operating condition and in the event of the occurrence of modelling errors in the control system.



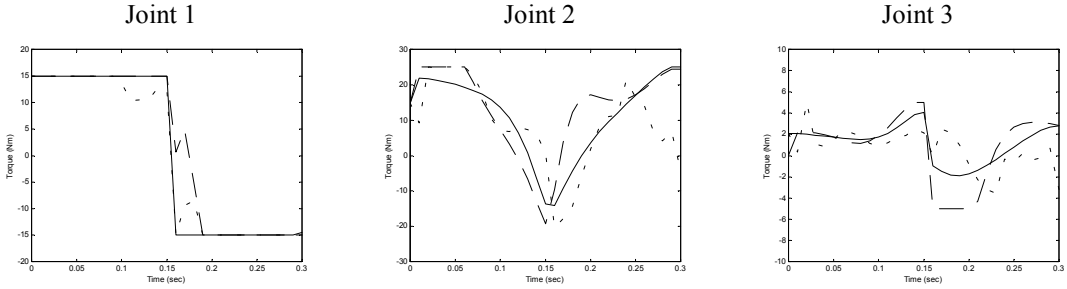
— Reference angle
--- Angle output - trajectory pre-shaping + PID controllers
… Angle output - PID + NN controllers

Fig. 7. Angular positions from each joint.



… Tracking error - trajectory pre-shaping + PID controllers
— Tracking error - PID + NN controllers

Fig. 8. Tracking errors from each joint.



— Open-loop torque
--- Closed-loop torque - trajectory pre-shaping + PID controllers
… Closed-loop torque - PID + NN controllers

Fig. 9. Open-loop and closed-loop torque on each joint.

Table 2. Summary of tracking error results.

Mass Loss (%)	Sum of Mean Absolute Tracking Errors (rad)		
	PID+KOH	PID+RBF	PID+TP
0	0.01864	0.02223	0.15540
10	0.02814	0.02536	0.17240
20	0.03539	0.03402	0.13546
30	0.04149	0.03976	0.12607
40	0.04663	0.04365	0.12690
50	0.05161	0.04750	0.12554

Although in all test cases, the PID and neural network controllers exhibit a very good performance, a significant increase in tracking errors can be observed as more mass is lost from the last link. However, this is to be expected since the neural network controllers are originally trained to cope with the robotic system which has been de-coupled into a set of de-coupled linear systems. As more mass is lost from the last link, the level of coupling in the overall system will increase. This will certainly lead to the deterioration in the performance of the neural network controllers. It can also be observed from the case where there is no mass loss from the robot arm that the sum of the mean absolute tracking errors over the trajectory when the extended Kohonen networks are used is slightly better than that when the radial-basis function networks are used. In contrast, once there are some modelling errors in the system, it can be seen that the tracking errors when the extended Kohonen networks are used are slightly higher than that when the radial-basis function networks are utilised. These results are caused by the differences in the network structure and the learning algorithm used. Recall that the output from a radial-basis function network is the weighted-sum of the signals from hidden neurons. This means that each control action of the radial-basis function network will be responsible by more than one neuron in the network. This also means that when a fault occurs in the control system, the fault tolerance load will be shared by a number of neurons. This makes the performance of the radial-basis function network remains high even when there is a large modelling error in the system. However, with the network structure like this, the learning process can also be a slightly difficult one since the target output has to be achieved through the distribution of the adjustment of a number of connection weights in the output layer. It means that the control action produced by the radial-basis function network can be worse than that produced by the network which the target network output is achieved through an easier approach of adjusting one connection weight in the output layer. This will be the case here since the learning algorithm used to adjust the extended Kohonen network involves the adjustment of the connection weight of a neuron in the motor map which can be treated as a connection weight in the output layer. This leads to the command tracking performance of the extended Kohonen network being better than that of the radial-basis function network in the normal operating condition. Note that the mentioned normal operating condition is also the condition at which the neural network controllers are subjected to the process of learning. Following the same line of reasoning, since the control action produced by the extended Kohonen network is the direct result from the firing of a neuron in the state map, the fault tolerance load will only be directed to one neuron in the state map and another neuron in the motor map. Without load sharing as in the case of the radial-basis function network, it comes to

no surprise that the fault tolerance performance of the extended Kohonen network would be worse than that of the radial-basis function network when modelling errors exist in the system.

6. Multi-Objective Optimisation Using a Genetic Algorithm

In practice, the maximum torque limits, which are used in the time-optimal trajectory calculation process for a closed-loop control, are usually less than the actual torque limits on the actuators. This safety precaution is done in order to allow some margins of error for possible discrepancies introduced to the system by modelling errors and controller dynamics.[4] This implies that for a given set of the actual torque limits of the actuators, there is a set of admissible torque limits combinations that can lead to a certain level of time-optimality within an acceptable range of tracking error. In addition, in certain applications such as welding or edge-deburring it is possible to modify the end-effector trajectory in Cartesian space without effecting the task requirement provided that the position and orientation of the work piece at which the end-effector has to remain in contact with can be modified accordingly. The control task discussed in section 3 is an example which reflects such applications. By modifying the initial and final locations of the straight-line path, the task description in the application viewpoint would remain the same while the angular trajectory at which the robot joint has to follow would be different. Such change in the angular trajectory would lead to a variation in the position tracking error. Combining with the issue on torque limits, this points to a design problem in robotic applications. The objective of such problem is to find a combination of torque limits from a set of admissible torque ranges and the initial and final position of the end-effector which will lead to a trajectory which meets the time-optimality and tracking error constraints. This is a multi-objective optimisation problem since it would be highly unlikely to obtain a single trajectory that can minimise both the trajectory time and tracking error simultaneously. A multi-objective genetic algorithm (MOGA) will be used to solve the problem associated with the torque limit and end-effector position selection in this study. The problem formation and the genetic operators used are discussed as follows.

6.1. *Decision Variables*

A 3-dof robot with the task of tracking a straight-line path in Cartesian space presented earlier is used to demonstrate this multi-objective optimisation problem. The decision variables of the problem consist of the torque limit combination and the initial and final position of the end-effector. Assuming that the magnitudes of the maximum and minimum torque limits are the same for each actuator, the torque limit part of the decision variables would consist of the magnitude of the torque limits of each joint. In this study, the range of the magnitudes of the torque limits on joints 1, 2 and 3 are set to 15-30, 25-40 and 5-20 Nm, respectively. The lower bounds of the limits (i.e. 15, 25, 5) are based on the maximum allowable trajectory time requirement of 0.3 seconds, while the upper bounds of the torque limits (i.e. 30, 40, 20) are set by the actual torque limits of the actuators.

Moving onto the part of decision variables which involves the positions of the end-effector. In order to create a fixed-length path in Cartesian space, two vectors are required: the position vector for the initial position of the end-effector and the direction vector pointing from the initial position toward the desired final position of the end-effector. This requirement can be achieved by setting up two search variables. The first variable will be the initial location of the end-effector while the second variable will be another point in the robot workspace at which a direction vector pointing from the initial position of the end-effector toward this point can be established. In this investigation the search range for the initial position of the end-effector is given by (0.721-0.751, 0.211-0.241, 0.078-0.108) in the x, y and z directions, respectively. In contrast, the search range for the location of the other point in the robot workspace is set to (-0.015-0.015, 0.839-0.869, 0.339-0.369) in the x, y and z directions, respectively. Note that the search ranges for these two points are in the vicinity of the initial and final positions of the straight-line path described earlier in section 3.

6.2. *Objective Variables*

There are two optimisation objective variables in this problem: the tracking error and the trajectory time objectives. The tracking error objective is expressed in terms of the sum of the mean absolute errors over three joints, calculated over the whole trajectory. The trajectory time objective is the optimal trajectory time obtained from the time-optimal control algorithm. Note that the sampling period used in the simulation of this 3-dof robotic closed-loop system is 0.01 seconds. Hence, the trajectory time will always be in the form of $0.01m$ where m is a positive integer.

6.3. *Chromosome Coding*

Nine decision variables – the magnitudes of the torque limits from all three joints and the co-ordinates along three axes of the two points for identifying the straight-line path – are concatenated together and coded to form a chromosome. Two chromosome coding schemes are explored here: Gray and integer-based coding schemes. The torque ranges for all three joints are discretised using a search step of 0.5 Nm. This leaves 31 search points for the magnitude of the torque limits of each joint. In a similar way, the search ranges of the co-ordinates of the two points for dictating the location of the straight-line path are discretised using a search step of 0.001 m. This also leaves 31 search points for the co-ordinate in each axis. With the use of a Gray coding scheme, a Gray code of length 5 can be used to represent a decision variable. The total length of the chromosome in this case would be equal to 45. Note that there are certain search points obtained after decoding the chromosome which lie outside the required search space. These points are mapped back into the feasible region by changing the most significant bit of the Gray code section representing the particular decision variable that violates the feasibility constraint into zero. In contrast to the case of the Gray coding scheme, with the use of an integer-based coding system a single gene can be used to represent a decision variable. Each gene can then take an

allele value from a set which is composed of 31 integers ranging from 0 to 30. The chromosome length in this case would be equal to nine.

6.4. *Fitness Assignment and Fitness Sharing*

The ranking method as described in Fonseca and Fleming[11] is used to rank each individual in the population. Following that, a linear fitness interpolation is used to assign fitness to each individual. Fitness sharing, with the use of triangular sharing function, is then carried out in normalised objective space.

6.5. *Selection Method*

Stochastic universal sampling[12] is used in the fitness selection. The elitist strategy used is to select two individuals with the highest fitness and pass onto the next generation without crossover or mutation.

6.6. *Crossover and Mutation Methods*

The standard one-point crossover is used in the recombination. Two individuals are allowed to perform crossover if, and only if, they are within the mating restriction distance from each other. For simplicity, the mating restriction radius is set to equal to the sharing radius and the consideration on the distance between the two individuals is also done in normalised objective space. For the case of chromosome coding using a Gray code, a standard bit-flipped operation is used for the mutation. In contrast, the value 1 will be added to or subtracted from the allele value of the mutated gene to achieve mutation in the integer-based coding system. The parameter settings for the MOGA are summarised in Table 3.

For the purpose of comparison, the random search technique is also used to find the Pareto optimal solutions in this study. Eschenauer *et al*.[13] have explained that in the case of a multi-objective optimisation, the random search method can generally be used to obtain a non-dominated solution set. In the random search technique, a set of random solutions is generated. Then non-dominated solutions are picked from this

Table 3. Parameter settings for the MOGA.

Parameter	Value
Chromosome length	
Gray code	45
Integer-based code	9
Crossover probability	0.8
Mutation probability	
Gray code	0.02
Integer-based code	0.1
Sharing and mating restriction radii	0.03
Population size	30
Number of elitist individuals	2
Number of generations	30

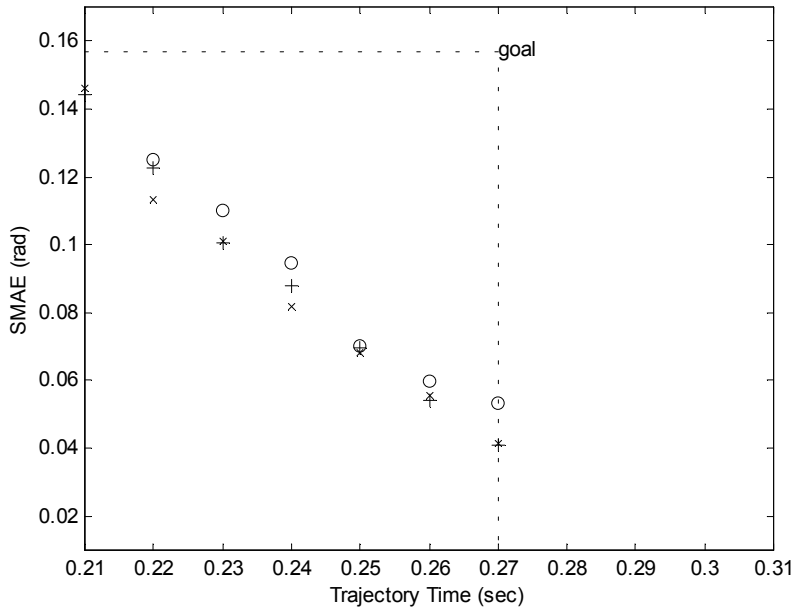
solution set. This can be done by applying the ranking mechanism used in the MOGA to the initial random solutions and selected solutions with the highest rank. Since a genetic algorithm also uses randomly generated solutions as its initial search points, the random search has already been embedded into the genetic algorithm as the initial search procedure. This means that a comparison between the non-dominated solutions found from the initial population of the genetic algorithm and the non-dominated solutions obtained from the last generation of the genetic algorithm run would provide an adequate comparison in terms of the comparison with the random search method. The description of the case studies explored, the simulation results and the discussions will be given in the next section.

7. Results from Using a Genetic Algorithm and Discussions

Two case studies are investigated in this paper. The aim of the first case study is to find a set of torque limit combinations and straight-line paths which lead to trajectories with the sum of the mean absolute tracking errors ≤ 0.15708 radians (3 degrees per joint) and the trajectory time ≤ 0.27 seconds. The aim of the second case study is to find a set of torque limit combinations and straight-line paths which lead to trajectories with the sum of the mean absolute tracking errors ≤ 0.07854 radians (1.5 degrees per joint) and the trajectory time ≤ 0.30 seconds. The purpose of the first case study is to find solutions that concentrate more on optimising the trajectory time while the second case study emphasises on the tracking error optimisation. The simulation results for these two cases are summarised in Figs. 10 and 11. Note that the displayed results are the combination of Pareto optimal solutions obtained from five different simulation runs. In addition, the initial populations used in both approaches of the MOGA in each simulation run are generated such that the resulting decision variables are the same. In other words, the initial populations used in the two approaches are equivalent in terms of the decision variables obtained after decoding the chromosomes.

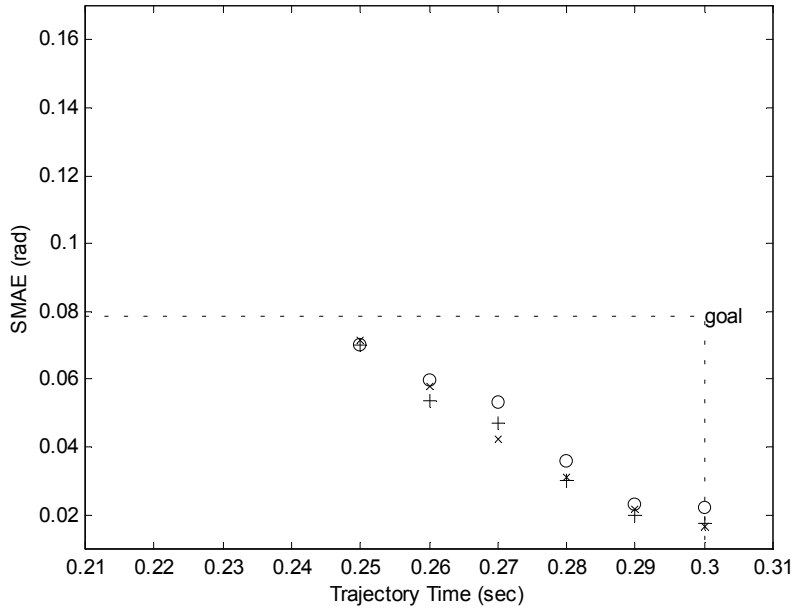
Prior to any analysis on the simulation results can be carried out, a number of points are required to be made clear. The Pareto front results are used to represent two main aims of the search; these are to find the range of variety in solutions and to locate the solutions which are close to the true Pareto optimal solutions of the problem. For this time-optimal control application, the exact range of variety in solutions is known. Such knowledge is gained by inspecting the non-dominated solutions and their corresponding objectives in the solution set itself. This statement will be made clearer later on in the discussions. Nonetheless, similar to the majority of engineering applications, the theoretical, or true, Pareto optimal solutions of the problem are not known. Of course, there will be a possibility that some of the Pareto optimal solutions found by one technique can be dominated by the solutions found by another technique. In order to compare the Pareto optimal solutions obtained from each technique objectively, both points of view on the variety in solutions found and the number of solutions found which cannot be dominated by the solutions obtained from other techniques need to be considered.

First of all, consideration is placed on the simulation results from the first case study. Both the MOGA with a Gray coding scheme and the MOGA with an integer-based coding scheme can locate seven distinct solutions while the random search fails



× MOGA with a Gray coding scheme
\+ MOGA with an integer-based coding scheme
○ Random search

Fig. 10. Pareto optimal solutions from case 1.



× MOGA with a Gray coding scheme
\+ MOGA with an integer-based coding scheme
○ Random search

Fig. 11. Pareto optimal solutions from case 2.

to locate a solution with the trajectory time of 0.21 seconds. For this case study, there can be only seven distinct solutions in the Pareto optimal solution set. This is because the solution that has a trajectory time of 0.21 seconds and still has the tracking error within the target value is obtained for magnitudes of torque limits which are close to the actual limits on the actuator torque. In addition, there are only seven distinct solutions which can occupy the trajectory time solution space from $t = 0.21$ seconds to $t = 0.27$ seconds with an increment of 0.01 seconds (the sampling period). As far as the variety of solutions found is concerned, both approaches of the MOGA are equally good in this respect. With a close inspection, it is noticeable that all solutions found by both approaches of the MOGA dominate all optimal solutions found by the random search. However, after comparing the results found by both approaches of the MOGA, it is found that the solutions with the trajectory times of 0.21, 0.23, 0.26 and 0.27 seconds found by the MOGA with a Gray coding scheme are dominated by the corresponding solutions found by the MOGA with an integer-based coding scheme. At the same time, the solutions found by the MOGA with an integer-based coding scheme which have trajectory times of 0.22, 0.24 and 0.25 seconds are dominated by the solutions obtained by the MOGA with a Gray coding scheme. In this respect, it can be said that the search performances of the two MOGA approaches are very close to one another.

Moving onto the second case study: all three search techniques are capable of locating six distinct solutions. Note that for this case study, there can be a maximum of six distinct solutions in the Pareto optimal solution set. This is concluded from the results obtained from the first case study which indicates that the solution which has the minimum allowable trajectory time and also has the tracking error which is smaller than 0.07854 radians is the one with the trajectory time of 0.25 seconds. With the maximum allowable trajectory time being limited to 0.3 seconds by the search target and the sampling period is set to 0.01 seconds, there are only six distinct solutions with the trajectory times ranging from 0.25 to 0.30 seconds that can cover the whole Pareto front. The simulation results in this case study also reveals that all solutions found by the MOGA with an integer-based coding scheme dominates all solutions found by the random search. In contrast, the MOGA with a Gray coding scheme can only find five solutions which dominate the solutions located by the random search. The only solution found by the MOGA with a Gray coding scheme which is dominated by the solution found by the random search is the one with the trajectory time of 0.25 seconds. Among the solutions found by the two MOGA approaches, two solutions found by the MOGA with a Gray coding scheme dominates the solutions located by the MOGA with an integer-based coding scheme. These two solutions are the solutions with the trajectory times of 0.27 and 0.30 seconds. In contrast, the MOGA with an integer-based coding scheme can locate four distinct solutions that dominates the solutions found by the MOGA with a Gray coding scheme: the solutions with the trajectory times of 0.25, 0.26, 0.28 and 0.29 seconds. In overall, it can be noticed that the performance of the MOGA with an integer-based coding scheme is slightly higher than that of the MOGA with a Gray coding scheme.

It can be seen from the results that the MOGA with an integer-based coding scheme has emerged as the most effective method in finding the Pareto front for this problem. This conclusion is supported by both viewpoints on the variety of solutions found and the number of found solutions which cannot be dominated by solutions

obtained from the other techniques. In addition, two further observations can be made from the results in both case studies. Firstly consider the solutions at the middle of the Pareto fronts; these are the solutions with the trajectory times of 0.24 seconds and 0.27 seconds from the first and the second case studies, respectively. At these locations, it can be seen that the solutions that are located by the MOGA with an integer-based coding scheme are clearly dominated by those identified by the MOGA with a Gray coding scheme. Moreover, the differences between the solutions found by the two MOGA approaches are more observable at the middle locations than at the extremes of the Pareto fronts. This observation is further illustrated in Tables 4 and 5 where the differences between the solutions, located by both approaches of the MOGA, at the extremes and middle of the Pareto fronts are displayed. Although this observation does not supersede the conclusion regarding the suitability of the integer-based coding scheme, further analysis and simulations are recommended to confirm whether this is an isolated incident or it can also be detected in other cases. The second important point, which can be observed from both case studies, is that nearly all of the solutions found by the random search method cannot dominate the solutions found by both approaches of the MOGA. Since the solutions found by the random search method in this case are the non-dominated solutions from the initial population of the genetic algorithm, this indicates that successful evolution has been accomplished by the MOGA.

Table 4. Solutions at the extremes and middle of the Pareto fronts from case 1.

Coding	Decision Variables									Objectives	
Scheme	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	t	SMAE
Gray	30.0	40.0	14.0	0.735	0.214	0.106	-0.005	0.849	0.343	0.21	0.14613
Integer	29.5	40.0	18.0	0.744	0.218	0.108	-0.015	0.843	0.349	0.21	0.14403
Gray	25.0	35.0	15.0	0.750	0.219	0.078	-0.005	0.867	0.354	**0.24**	**0.08179**
Integer	24.0	40.0	19.0	0.748	0.219	0.078	0.013	0.846	0.369	**0.24**	**0.08789**
Gray	19.5	35.0	11.0	0.748	0.227	0.081	0.008	0.859	0.341	0.27	0.04156
Integer	19.5	34.5	12.0	0.724	0.236	0.078	0.013	0.869	0.359	0.27	0.04084

T_i - Magnitude of torque limits on joint i (Nm)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)
t - Trajectory time (second)
SMAE - Sum of mean absolute tracking errors (rad)

Table 5. Solutions at the extremes and middle of the Pareto fronts from case 2.

Coding	Decision Variables									Objectives	
Scheme	T_1	T_2	T_3	x_{ini}	y_{ini}	z_{ini}	x_{fin}	y_{fin}	z_{fin}	t	SMAE
Gray	22.5	39.5	17.0	0.739	0.235	0.081	-0.010	0.866	0.359	0.25	0.07145
Integer	23.0	38.5	16.5	0.751	0.213	0.079	0.015	0.869	0.361	0.25	0.06986
Gray	19.5	34.0	8.5	0.747	0.223	0.081	0.006	0.868	0.346	**0.27**	**0.04235**
Integer	20.0	35.0	17.0	0.747	0.231	0.087	0.013	0.866	0.345	**0.27**	**0.04686**
Gray	15.0	40.0	6.0	0.730	0.237	0.104	-0.015	0.862	0.367	0.30	0.01649
Integer	15.0	34.5	10.0	0.737	0.222	0.092	-0.013	0.863	0.351	0.30	0.01760

T_i - Magnitude of torque limits on joint i (Nm)
$(x_{ini}, y_{ini}, z_{ini})$ - Initial position (m)
$(x_{fin}, y_{fin}, z_{fin})$ - Required second position (m)
t - Trajectory time (second)
SMAE - Sum of mean absolute tracking errors (rad)

8. Conclusions

In this paper, the robotic application which is chosen to illustrate the effectiveness in combining neural networks and a genetic algorithm at the application task level is a time-optimal control application. The task of tracking a straight-line path in Cartesian space is given to the robot in this case. The time-optimal joint trajectory time history is calculated by using the time-optimal control algorithm as described by Shiller and Lu.[2] Time-optimality is achieved by executing a bang-bang control, where the control torque signal in one joint is saturated and the control torque signal on other joints is adjusted accordingly such that the torque limits on each actuator are not violated. However, the trajectory time history obtained from the time-optimal control algorithm is calculated by using only the open-loop dynamics of the robot model. Previously, in order for the trajectory to be used as input to the control loop, either the time history had to be pre-shaped[4] or the inclusion of radial-basis function network controllers, which are trained using feedback error learning, is required.[7] In this paper, the use of extended Kohonen networks, which are trained using reinforcement learning, has been proven to be an effective method in compensating for the closed-loop dynamics and modelling errors. This results in being able to use the trajectory time history as the input to the control loop directly without the use of trajectory pre-shaping scheme. In addition, the extended Kohonen network is also proven to be more efficient than the radial-basis function network when the robot arm is operated in the normal operating condition. However, one drawback is that the fault tolerance capability of the extended Kohonen network is slightly lower than that of the radial-basis function network.

Subsequently, a genetic algorithm has been used to solve a multi-objective optimisation involving the selection of torque limits and an end-effector path subject to time-optimality and tracking error constraints. Two approaches of a multi-objective genetic algorithm (MOGA) have been used in this application: the MOGA with a Gray coding scheme and the MOGA with an integer-based coding scheme. The simulation results suggest that the integer-based chromosome is more suitable than the Gray chromosome at representing the decision variables. This makes the MOGA with an integer-based coding scheme emerge as the most effective method in finding the Pareto optimal solutions for this problem.

Acknowledgement

This work was supported by the Thailand Research Fund (TRF) under the Post-Doctoral Research Fellowship Programme (Grant Number: PDF/05/2543).

References

1. J. E. Bobrow, S. Dubowsky and J. S. Gibson, Time-optimal control of robotic manipulators along specified paths, *Int. J. Robotics Research* **4**(3) (1985) 3-17.
2. Z. Shiller and H.-H. Lu, Computation of path constrained time optimal motions with dynamic singularities, *ASME J. Dyn. Syst. Meas. & Control* **114**(1) (1992) 34-40.
3. G. Sahar and J. M. Hollerbach, Planning of minimum-time trajectories for robot arms, *Int. J. Robotics Research* **5**(3) (1986) 90-100.

4. Z. Shiller, H. Chang and V. Wong, The practical implementation of time-optimal control for robotic manipulators, *Robotics and Computer-Integrated Manufacturing* **12**(1) (1996) 29-39.
5. Z. Shiller, Time-energy optimal control of articulated systems with geometric path constraints, *ASME J. Dyn. Syst. Meas. & Control* **118**(1) (1996) 139-143.
6. M. Kawato, Y. Uno, M. Isobe and R. Suzuki, Hierarchical neural network model for voluntary movement with application to robotics, *IEE Control Syst. Mag.* **8**(2) (1988) 8-16.
7. N. Chaiyaratana and A. M. S. Zalzala, Hybridisation of neural networks and genetic algorithms for time-optimal control, *Congress on Evolutionary Computation'99* **1** (1999) 389-396.
8. C. M. Fonseca and P. J. Fleming, Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, *Proc. Fifth Int. Conf. Genetic Algorithms* (1993) 416-423.
9. E. Freund, Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators, *Int. J. Robotics Research* **1**(1) (1982) 65-78.
10. T. M. Martinetz, H. J. Ritter and K. J. Schulten, Three-dimensional neural net for learning visuomotor coordination of a robot arm, *IEEE Trans. Neural Networks* **1**(1) (1990) 131-136.
11. C. M. Fonseca and P. J. Fleming, Multiobjective genetic algorithms made easy: Selection, sharing and mating restriction, *Proc. Second Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications* (1995) 45-52.
12. J. E. Baker, *An Analysis of the Effects of Selection in Genetic Algorithms* (Ph.D. Thesis, Vanderbilt University, Tennessee, 1989).
13. H. Eschenauer, J. Koski and A. Osyczka (Eds.), *Multicriteria Design Optimization: Procedures and Applications* (Springer-Verlag, Berlin, 1990).