HYBRIDISATION OF NEURAL NETWORKS AND GENETIC ALGORITHMS FOR TIME-OPTIMAL CONTROL

Nachol Chaiyaratana
Department of Production Engineering
King Mongkut's Institute of Technology North Bangkok
1518 Piboolsongkram Road
Bangkok, THAILAND 10800
nachol1@go.com

Ali M. S. Zalzala[1]
Department of Computing and Electrical Engineering
Heriot-Watt University
Edinburgh, UK, EH14 4AS
A.Zalzala@hw.ac.uk

Abstract- This paper presents the use of neural networks and genetic algorithms in time-optimal control of a closed-loop robotic system. Radial-basis function networks are used in conjunction with PID controllers in an independent joint position control to reduce tracking errors. The results indicate that using neural network controllers is more effective than using trajectory pre-shaping scheme, reported in early literature. Subsequently, a genetic algorithm with a weighted-sum approach and a Multi-Objective Genetic Algorithm (MOGA) are used to solve a multi-objective optimisation problem related to time-optimal control. The results indicate that the MOGA is the best method in terms of the Pareto front coverage while the genetic algorithm with a weighted-sum approach is more effective in terms of finding the best individual according to the weighted-sum criteria. As a result of using both neural networks and genetic algorithms in this application, an idea of a task hybridisation between neural networks and genetic algorithms for use in a control system is also effectively demonstrated.

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 by Bobrow et al. (1985). Over the years, this algorithm has undergone a number of refinements and one of the latest modifications has been described in Shiller and Lu (1992). 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 (Sahar and Hollerbach, 1986).

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. (1996) 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. Two 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 (Shiller, 1996). 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 (Shiller et al., 1996).

In this paper, time-optimal control is used in a closed-loop robotic system. This is done in a similar way to that described in Shiller et al. (1996) except that the actuator dynamics are not considered. However, neural network controllers are used in conjunction with standard controllers, which leads to the redundancy of the use of trajectory pre-shaping. In addition, a further multi-objective optimisation problem associated with the use of time-optimal control in a feedback system is also considered. In this case, the decision variables consist of the magnitude of the torque limits within the actuator on each robot joint while the objective variables are the trajectory time and the position tracking error. Two approaches on multi-objective optimisation using a genetic algorithm, namely a genetic algorithm with a weighted-sum approach and a Multi-Objective Genetic Algorithm (MOGA) (Fonseca and Fleming, 1993), have been used to solve this optimisation problem. Furthermore, a hill-climbing method and a random search have also been used to solve the same problem in order to compare with the genetic algorithm performances. Since neural networks and genetic algorithms are used in the different part of the

[1] Corresponding author

control application, in essence this indicates a task hybridisation between neural networks and genetic algorithms.

The paper is presented as follows. The time-optimal control algorithm as described by Shiller and Lu (1992) is briefly explained in section 2. In addition, the trajectory pre-shaping scheme is also explained in this section. In section 3, the control structure of the robotic system and the improvement in the system performance gained by using neural network controllers is discussed. The multi-objective optimisation problem associated with time-optimal control and the optimisation techniques used to solve the problem is explained in section 4. Simulation results from using different optimisation techniques are shown in section 5. Discussions on the optimisation results are given in section 6. Finally, conclusions are given in section 7.

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

In summary, time-optimal control algorithm as described by Shiller and Lu (1992) 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 Figure 1.

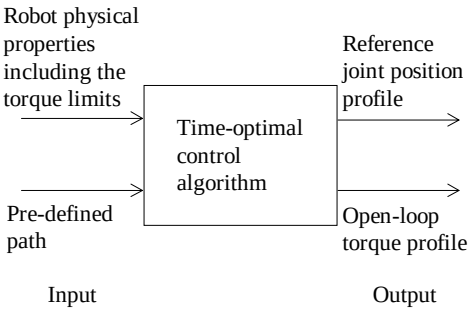

Figure 1: Schematic diagram of the time-optimal control algorithm

In Figure 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.

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 (Shiller, 1996; Shiller et al., 1996). In order to solve the problem, Shiller et al. (1996) 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 has been reported in early literature, it will be demonstrated in the following section that the use of neural networks to compensate for dynamics of controllers and modelling errors helps to remove the need of using trajectory pre-shaping.

3 Control Structure and Neural Network Contribution

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

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

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 independent joint 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 (1982) has suggested such a control signal which takes the form of

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

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 equation (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 relationships 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 as schematically displayed in Figure 2(a). 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 Figure 2(b).

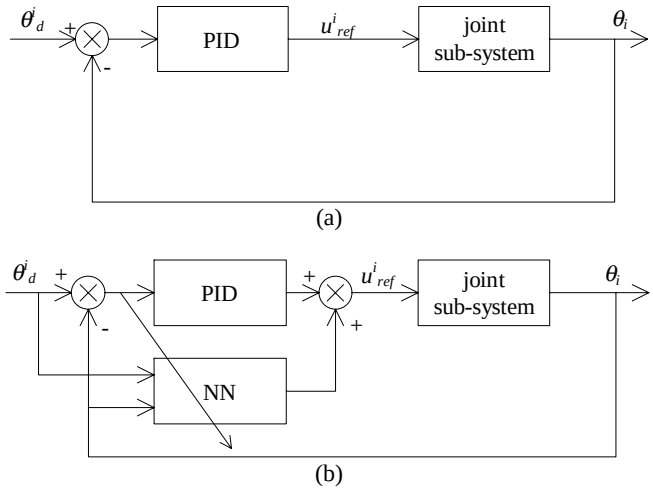

Figure 2(a): A joint sub-system of the robot system using independent joint control in conjunction with PID controllers

Figure 2(b): Neural network and PID controllers in each joint control loop

However, with the arrangement shown in Figure 2(b), it is not possible to derive an exact desired neural network output training signal for use in supervised learning. An alternative training signal must be acquired. In a standard supervised learning, the difference between the actual neural network output and the desired neural network output - the training error - is used to update the network parameters such as the connection weights. In the case where a neural network is placed in a closed-control loop like in Figure 2(b), the feedback error signal can be used as the alternative to the training error signal. This modified supervised learning scheme is called feedback error learning (Kawato et al., 1988).

In this study, radial-basis function networks are used to assist PID controllers in the position control loop. Position feedback error learning is used to train the connection weights while the centres of the Gaussian radial-basis functions are unsupervisedly trained. The training algorithm used is similar to the one described by Fritzke (1994) except no new centres are incremented into the network during training and the nearest neighbour centre concept is used instead of the topological neighbour neuron concept. Three neural network controllers, one for each joint sub-system, are trained and tested for use in position control of a 3-dof robot by using a combination of the time-optimal position trajectory and its corresponding position feedback as both the training and the testing samples. Note that this time-optimal trajectory is obtained for a robot task of tracking a straight line path in Cartesian space with the torque limits on joints 1, 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 for the case of PID controllers with trajectory pre-shaping and the case of PID and neural network controllers are shown in Figures 3, 4 and 5.

Parameter	Value
Number of Gaussian radial-basis functions in each network	30
Number of connection weights in each network	30
Number of input nodes in each network	2
Number of output nodes in each network	1
Learning rate parameter (weight training)	0.001
Number of training samples	30
Number of training epochs	200

Table 1: Parameter settings for training neural networks

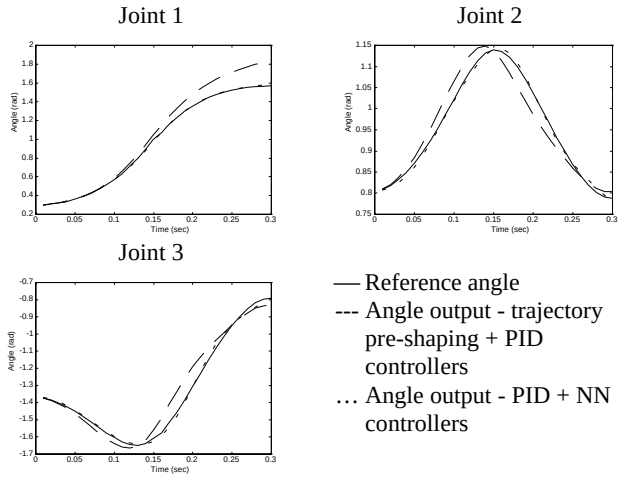

Figure 3: Angular positions from each joint

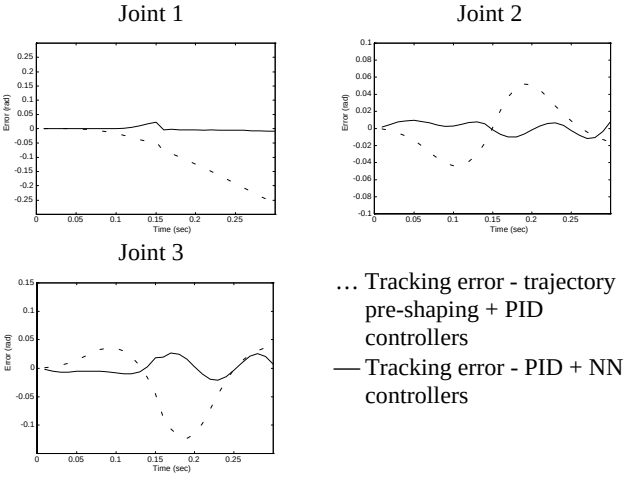

Figure 4: Tracking errors from each joint

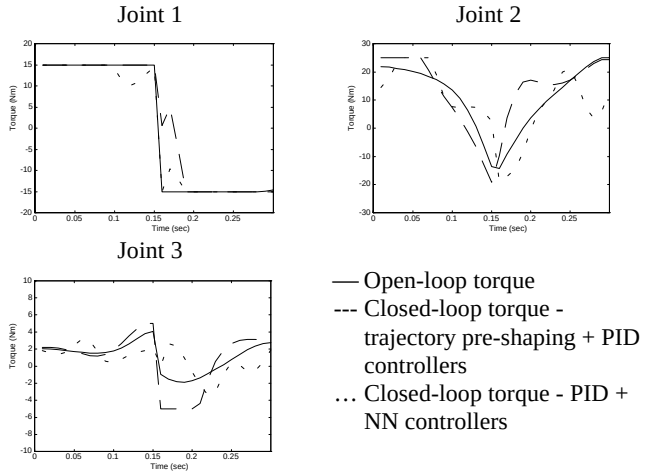

Figure 5: Open-loop and closed-loop torque on each joint

In Figures 3 and 4, the simulation results indicate that with the use of the neural 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 Figure 5, with the use of neural network controllers, the characteristics of the closed-loop torque profiles are similar to those of open-loop control. This indicates that time-optimality has been achieved within the torque constraints. Note that these trained radial-basis function 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 neural 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 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 squared position tracking error and the sum of the absolute error, 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 in the same situations are also given for comparison purposes.

Mass Loss (%)	Squared Error (rad^2)		Absolute Error (rad)	
	PID + NN	PID + TP	PID + NN	PID + TP
0	0.0084	0.5894	0.6669	4.6620
10	0.0103	0.6350	0.7607	5.1721
20	0.0205	0.3191	1.0206	4.0637
30	0.0304	0.2754	1.1929	3.7820
40	0.0365	0.2656	1.3096	3.8071
50	0.0409	0.2509	1.4249	3.7661

PID + NN - PID and neural network controllers
PID + TP - PID controllers with trajectory pre-shaping

Table 2: Summary of tracking error results

Again, from Table 2 it is noticeable that in all test cases, the use of the neural networks as assistants to the 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.

4 Multi-Objective Optimisation Problem and Optimisation Techniques

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 actuator. 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 (Shiller et al., 1996). 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. This points to a design problem in robotic applications in which the objective is to find a combination of torque limits from a set of admissible torque ranges 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 the tracking error simultaneously. A Multi-Objective Genetic Algorithm (MOGA) (Fonseca and Fleming, 1993) and a genetic algorithm with a weighted-sum

approach have been used to solve the problem associated with the torque limits selection in this study. The problem formation and the genetic operators used are discussed as follows.

4.1 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. Assuming that the magnitudes of the maximum and the minimum torque limits are the same for each actuator, the decision variables of a possible solution 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.

4.2 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 value over three joints, calculated over the whole trajectory. The trajectory time objective is the optimal trajectory time obtained from the time-optimal control algorithm described in section 2. 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.

4.3 Chromosome Coding

Three decision variables - the magnitudes of the torque limits from all three joints - are concatenated together and coded using a Gray code to form a chromosome. The torque ranges of 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 which can be coded with a Gray code of length 5. The total length of the chromosome in this case is 15. 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.

4.4 Fitness Assignment and Fitness Sharing

The ranking method, as described in Fonseca and Fleming (1995), is used to rank each individual in the population in the case of the MOGA. Then, 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. In the case of the genetic algorithm with a weighted-sum approach, the objective functions are weighted and added together to form a single objective. Then fitness is calculated from the weighted-sum objective in the usual way for a single-objective minimisation problem. In effect, this will force the algorithm to concentrate the search on only one area of the Pareto front. For this reason, fitness sharing is not implemented in the genetic algorithm with a weighted-sum approach.

4.5 Selection Method

Stochastic universal sampling (Baker, 1989) 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.

4.6 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. In addition, a standard bit-flipped operation is used for the mutation. The parameter settings for both the MOGA and the genetic algorithm with a weighted-sum approach are summarised in Table 3.

Parameter	Value
Chromosome length	15
Crossover probability	0.8
Mutation probability	0.07
Sharing radius (MOGA only)	0.003
Mating restriction radius	0.003
Population size	30
Number of elitist individuals	2
Number of generations	30

Table 3: Parameter settings for the genetic algorithms

Since the techniques involved in the search mechanism used in both the genetic algorithm with a weighted-sum approach and the MOGA are quite different from one another, another two search techniques are also presented here for the purpose of comparison with each of the genetic algorithm approach. These two standard techniques are the hill-climbing and random search methods. A brief description of these two methods is given as follows.

As mentioned earlier, in the case of the genetic algorithm with a weighted-sum approach, a multi-objective optimisation is reduced into a single objective one. The hill-climbing method is also a single-objective optimisation method which can be used to find an optimal solution

without the use of gradient information of the search space. The hill-climbing algorithm presented here is similar to the one reported in Mahfoud (1995), which is also recommended for use in the comparison with the genetic algorithm performance. The hill-climbing algorithm starts from an initial value of randomly chosen decision variables. Then the search cycle through the variables, trying perturbations on each one. A perturbation is either an upward or downward change in the value of a variable. The hill-climbing algorithm will take each improvement as it is found. After cycle through all the variables and no improvement is gained, the search step size is reduced by half and the search begins again. The hill-climbing algorithm will terminate when a search with the smallest step size yields no improvement.

Moving onto the random search technique: Eschenauer et al. (1990) 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 are generated. Then non-dominated solutions are picked from this 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.

By incorporating the weighted-sum criteria used in the genetic algorithm with a weighted-sum approach and the hill-climbing technique to the non-dominated solution sets obtained from the MOGA and the random search, another two best individual results can be obtained for comparison purposes. In brief, there will be a total of four best individual results, one from each technique. Note that the best individual result from the random search will be equivalent to the best individual from the first generation of a genetic algorithm run. On the other hand, by applying the ranking technique used in the MOGA to the last generation of the genetic algorithm with a weighted-sum approach, another non-dominated solution set can be obtained for comparison with the non-dominated solution sets obtained from the MOGA and random search. In total, there will be three sets of non-dominated solutions for comparison. Again, the non-dominated solution set from the random search will be equivalent to the non-dominated solution set in the initial population of the genetic algorithm. For simplicity, the same initial population is used in both the genetic algorithm with a weighted-sum approach and the MOGA for comparison purposes. The parameter settings for the hill-climbing and random search methods are summarised in Table 4.

Parameter	Value
Hill-Climbing Algorithm	
Initial search step size (Nm)	4.0
Final search step size (Nm)	0.5
Random Search	
Number of initial random solutions	30

Table 4: Parameter settings for hill-climbing and random search methods

5 Simulation Results

Two case studies are investigated in this paper. The aim of the first case study is to find a set of torque limits combinations which leads to trajectories with the sum of the mean absolute tracking error ≤ 0.10472 radians (2 degrees per joint) and the trajectory time ≤ 0.27 seconds. The aim of the second case study is to find a set of torque limits combinations which leads to trajectories with the sum of the mean absolute tracking error ≤ 0.05236 radians (1 degree 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 Tables 5-12. Note that more than one combination of torque limits can result in the same time-optimal trajectory.

6 Discussions

Inspection of the results from both case studies reveals that the weighted-sum solution found by the hill-climbing method is the best solution in the first case study and the worst solution in the second case study. In contrast, the genetic algorithm with a weighted-sum approach emerges as a more effective method where it is capable of locating the best weighted-sum solutions in both case studies. This highlights the problem associated with the nature of the hill-climbing algorithm. In the hill-climbing method, the search always starts from one guessed solution. In order for the hill-climbing method to find the globally optimal solution, the globally optimal solution has to be reachable from the initial solution. This means that the search path selected by the search algorithm must not lead to locally optimal solutions. This makes the selection of the initial solution a critical factor to the possibility of a successful search. Along the same lines of reasoning, the reason for the genetic algorithm with a weighted-sum approach being a more effective method for finding the best solution lies on the use of its parallel search mechanism. In the genetic algorithm, the search is initialised by a number of guessed solutions, this makes the selection of the initial solutions become less critical.

Decision Variables			Objective Val.	
T_1	T_2	T_3	Error	t
25.5	39.0	10.5	0.06931	0.23
24.0	34.0	12.0	0.05848	0.24
22.0	36.5	10.5	0.04251	0.25
20.5	30.5	9.0	0.04158	0.26
19.0	39.0	12.0	0.03659	0.27

Table 5: Pareto optimal solutions - random search (case 1)

Decision Variables			Objective Val.	
T_1	T_2	T_3	Error	t
27.5	40.0	18.5	0.08595	0.22
26.0	40.0	18.5	0.06129	0.23
23.5	37.5	20.0	0.05302	0.24
22.0	36.5	11.0	0.04251	0.25
20.0	31.5	11.0	0.03674	0.26
18.5	33.0	16.0	0.03044	0.27

Table 6: Pareto optimal solutions - WSGA (case 1)

Decision Variables			Objective Val.	
T_1	T_2	T_3	Error	t
29.0	40.0	11.0	0.08581	0.22
26.0	38.0	14.5	0.06376	0.23
24.0	38.5	19.0	0.05264	0.24
22.0	36.5	11.0	0.04251	0.25
20.0	38.0	16.0	0.03465	0.26
20.0	38.0	17.5	0.03465	0.26
18.5	34.0	9.5	0.03038	0.27

Table 7: Pareto optimal solutions - MOGA (case 1)

Approach	Pareto Front	Best Individual					
	Number of Distinct	Decision Variable			Objective Value		Weighted-Sum
	Solutions	T_1	T_2	T_3	Error	t	Objective Value
Hill-Climbing	-	26.0	40.0	20.0	0.06129	0.23	0.29129
Random Search	5	22.0	36.5	10.5	0.04251	0.25	0.29251
WSGA	6	26.0	40.0	18.5	0.06129	0.23	0.29129
MOGA	6	22.0	36.5	11.0	0.04251	0.25	0.29251

Table 8: Summary of results from the four approaches - weight on the mean absolute tracking error objective = 1.0 and weight on the trajectory time objective = 1.0 (case 1)

Decision Variables			Objective Val.	
T_1	T_2	T_3	Error	t
22.0	36.5	10.5	0.04251	0.25
20.5	30.5	9.0	0.04158	0.26
19.0	39.0	12.0	0.03659	0.27
17.5	27.5	13.5	0.02814	0.28
15.0	30.5	8.5	0.01976	0.30
15.0	34.0	14.0	0.01976	0.30
15.0	39.0	15.5	0.01976	0.30

Table 9: Pareto optimal solutions - random search (case 2)

Decision Variables			Objective Val.	
T_1	T_2	T_3	Error	t
22.0	38.5	5.5	0.04465	0.25
18.5	36.0	12.5	0.03038	0.27
16.0	35.5	7.0	0.02246	0.29
16.0	39.0	11.5	0.02246	0.29
15.0	27.0	19.5	0.01970	0.30

Table 10: Pareto optimal solutions - WSGA (case 2)

Decision Variables			Objective Val.	
T_1	T_2	T_3	Error	t
22.0	38.0	18.0	0.04248	0.25
22.0	38.5	17.5	0.04248	0.25
20.0	36.5	16.0	0.03465	0.26
20.0	38.0	17.0	0.03465	0.26
20.0	38.0	17.5	0.03465	0.26
18.5	32.5	9.0	0.03056	0.27
17.5	27.5	5.0	0.02814	0.28
16.0	35.0	16.0	0.02246	0.29
16.0	39.0	7.0	0.02246	0.29
15.0	28.5	18.5	0.01971	0.30
15.0	28.5	19.0	0.01971	0.30

Table 11: Pareto optimal solutions - MOGA (case 2)

Approach	Pareto Front	Best Individual					
	Number of Distinct	Decision Variable			Objective Value		Weighted-Sum
	Solutions	T_1	T_2	T_3	Error	t	Objective Value
Hill-Climbing	-	15.0	40.0	20.0	0.01976	0.30	0.2276
Random Search	5	15.0	30.5	8.5	0.01976	0.30	0.2276
WSGA	4	15.0	27.0	19.5	0.01970	0.30	0.2270
MOGA	6	15.0	28.5	18.5	0.01971	0.30	0.2271

Table 12: Summary of results from the four approaches - weight on the mean absolute tracking error objective = 10.0 and weight on the trajectory time objective = 0.1 (case 2)

In the above tables, T_1, T_2 and T_3 are the magnitudes of the torque limits (Nm) on joints 1, 2 and 3, respectively. The heading "Error" represents the sum of the mean absolute tracking error (rad) over three joints, calculated over the trajectory and t denotes the trajectory time (sec).

In other words, during the search, the possibility that some individuals in the population of the genetic algorithm to escape the attraction of local optimums is higher than that of a single solution in the hill-climbing method.

Overall, it can be seen that the MOGA has emerged as the most effective method in finding the Pareto front for this problem. This conclusion is supported by both view points on the variety of solutions found and the number of found solutions which cannot be dominated by solutions obtained from the other techniques. Another important point, which can be observed from both case studies, is that the solutions found by the random search method cannot dominate any solutions found by the MOGA. Since the solutions found by the random search method in this case are the non-dominated solutions of the initial population of the genetic algorithm, this indicates that successful evolution has been accomplished by the MOGA.

7 Conclusions

In this paper, the robotic application which is chosen to illustrate the effectiveness in combining neural networks and genetic algorithms at the application task level is a time-optimal control application. The task of tracking a straight line trajectory 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 (1992). 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 motion control algorithm is calculated by using only the open-loop dynamics of the robot model. Previously, in order for this trajectory time history to be used as input to the position control loop, the time history had to be modified using trajectory pre-shaping scheme (Shiller et al., 1996). In this paper, the use of neural networks as assistants to PID controllers 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 complicate pre-shaping, which has not been possible before in earlier literature.

Subsequently, genetic algorithms have been used to solve a multi-objective optimisation involving the selection of torque limits subject to time-optimality and tracking error constraints. Two approaches of multi-objective optimisation using a genetic algorithm have been used in this application: the genetic algorithm with a weighted-sum approach and the MOGA. The simulation results suggest that due to the local search nature of the genetic algorithm with a weighted-sum approach, it can produce a better weighted-sum individual than the MOGA. However, the MOGA has been proven to be a method which can produce a better non-dominated solution set. This confirms the nature of the MOGA of being a global search technique.

Bibliography

Baker, J. E. (1989). *An analysis of the effects of selection in genetic algorithms*, Ph.D. Thesis, Computer Science Department, Vanderbilt University, Nashville, TN.

Bobrow, J. E., Dubowsky, S. and Gibson, J. S. (1985). Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research*, *4*(3), 3-17.

Eschenauer, H., Koski, J. and Osyczka, A. (Eds.) (1990). *Multicriteria design optimization: Procedures and applications*. Berlin, Germany: Springer-Verlag.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Genetic Algorithms: Proceedings of the 5th International Conference*, Urbana-Champaign, IL, 416-423.

Fonseca, C. M. and Fleming, P. J. (1995). Multiobjective genetic algorithms made easy: Selection, sharing and mating restriction. *The 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, Sheffield, UK, 45-52.

Freund, E. (1982). Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators. *International Journal of Robotics Research*, *1*(1), 65-78.

Fritzke, B. (1994). Growing cell structures - A self-organizing network for unsupervised and supervised learning. *Neural Networks*, *7*(9), 1441-1460.

Kawato, M., Uno, Y., Isobe, M. and Suzuki, R. (1988). Hierarchical neural network model for voluntary movement with application to robotics. *IEE Control Systems Magazine*, *8*(2), 8-16.

Mahfoud, S. W. (1995). A comparison of parallel and sequential niching methods. *Proceedings of the 6th International Conference on Genetic Algorithms*, Pittsburgh, PA, 136-143.

Sahar, G. and Hollerbach, J. M. (1986). Planning of minimum-time trajectories for robot arms. *International Journal of Robotics Research*, *5*(3), 90-100.

Shiller, Z. (1996). Time-energy optimal control of articulated systems with geometric path constraints. *Transactions of the ASME. Journal of Dynamic Systems, Measurement, and Control*, *118*(1), 139-143.

Shiller, Z., Chang, H. and Wong, V. (1996). The practical implementation of time-optimal control for robotic manipulators. *Robotics and Computer-Integrated Manufacturing*, *12*(1), 29-39.

Shiller, Z. and Lu, H.-H. (1992). Computation of path constrained time optimal motions with dynamic singularities. *Transactions of the ASME. Journal of Dynamic Systems, Measurement, and Control*, *114*(1), 34-40.