A Dynamic Programming Approach to Redundancy Resolution with Multiple Criteria

A. Guigue, M. Ahmadi, M.J.D. Hayes, R. Langlois and F. C. Tang

Abstract— **This paper addresses the problem of generating optimal joint trajectories for redundant manipulators when multiple criteria are to be considered. A dynamic programming approach is proposed to generate the Pareto optimal solutions without having to deal with the shortcomings of the traditional weighting method. The two approaches are implemented on the model of a 7-DOF redundant manipulator with the end-effector moving along a prescribed trajectory, while the joint trajectories are required to minimize two performance criteria. The results prove that the dynamic programming approach provides a better approximation of the Pareto optimal set and more flexible and predictable framework to control the objective vectors.**

I. INTRODUCTION

By definition, a manipulator is said to be redundant when it possesses more degrees of freedom than those required to execute a prescribed task. As a result, the number of possible joint trajectories performing this task is in general infinite. Redundancy resolution is the process of selecting one of these solutions that optimize a performance criterion. The optimization problems arising from redundancy resolution involve functions of robot variables such as joint variables or joint torques. Therefore, they have naturally been formulated within the framework of optimal control theory for continuous-time systems or calculus of variations. Since the introduction of the optimal control theory [8] and calculus of variations [6] for redundancy resolution, the research has primarily focused on the inclusion of different types of constraints and development of effective numerical algorithms. More recently, an optimal control formulation has been proposed, taking joint torques as the control inputs and the joint torque limits, end-effector path, and workspace obstacles as constraints [5]. A variational formulation is used in [9] with kinematic compliant constraints and a numerical algorithm involving Newton iterations on discretized Lagrange function. These work mainly deal with a single criterion or a linear combination of multiple criteria.

With any complex application, naturally come several performance criteria that need to be considered concurrently within a single optimization problem. The most widely used method is the weighting method [6], [9], which consists of combining linearly the criteria transforming the problem into a single criterion problem. Hence, the latest developments in single criterion redundancy resolution could be utilized for multiple criteria redundancy resolution. This approach suffers from the fact that, in general, one cannot predict the variation of the performance criteria for the optimal solution as the weights vary. The shortcomings of this approach are illustrated in this paper for the model of the 7-DOF redundant robot which is part of the Captive Trajectory Simulation (CTS) system, to be installed inside a supersonic wind tunnel. Similar problems have been reported in the multiple criteria research community [4].

In this paper, we propose a dynamic programming approach to redundancy resolution with multiple criteria which handles the multiple criteria problem without transforming it into a single criterion problem. Dynamic Programming [2] allows the implementation of Pareto Optimality which defines the optimal solution to a multiple criteria problem. In the process of dynamic programming, all the Pareto Optimal solutions can be generated which we can later choose a solution from based on a preference. The approach has been successfully implemented on the CTS robot model which will follow in later sections. Two criteria have been used for this study, but the approach is very promising for this application where there are several critical criteria and constraints but only one degree of redundancy available.

This paper is organized as follows. Section 2 describes the Captive Trajectory Simulation (CTS) experiments and the redundant robotic platform for which a task with two criteria is derived. Section 3 reviews the variational formulation with linearly combined criteria and highlights important drawbacks associated with the weighting method. Section 4 starts by proposing the dynamic programming approach with the weighting method and shows how the results compare to those of variational formulation. Then the approach is expanded to generate Pareto Optimal solutions and results presented. Section 5 concludes the findings and propose future work.

II. PROBLEM DEFINITION

The 7-DOF robot used for this study is modeled after a Captive Trajectory Simulation (CTS) system with the objective of emulating the store (any object released from an aircraft) trajectory. This robot as illustrated in Figure 1, operating within a supersonic wind-tunnel, holds at the end-effector the model of a store mounted on a sensitive

A. Guigue, M. Ahmadi, M.J.D. Hayes and R. Langlois are with the Mechanical and Aerospace Engineering (MAE) Department, Carleton University, Ottawa, Canada aguigue@connect.carleton.ca, mahmadi@mae.carleton.ca, jhayes@mae.carleton.ca and rlangloi@mae.carleton.ca

F. C. Tang is with the Institute for Aerospace Research (IAR) of National Research Council Canada (NRC) Norm.Tang@nrc-cnrc.gc.ca

force sensor. The significance of this setup is its ability in reproducing the scaled version of aerodynamic loads acting on the store in the vicinity of the aircraft model, where the aerodynamic interference is extremely complex and almost impossible to predict. The presence of manipulator in this area is undesirable as it disturbs the flow properties and can also impose additional disturbing loads on the robot links. Therefore, it is desirable to have the body of the robot as far as possible from the interference region. This is possible by keeping one of the aerodynamically-shaped upper links (Gooseneck) as vertical as possible when the robot operates underneath the wing of the airplane model. This constitutes one of the two criteria used in this study, the other being joint speed norm. Details about this system and CTS testing can be found in [1].

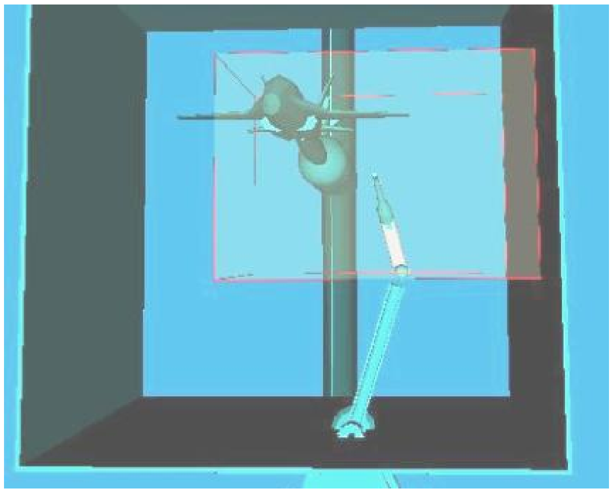

Fig. 1. The CTS manipulator inside the wind tunnel.

The robot has a box-shaped 6-DOF task space as illustrated in Figure 1. An attractive characteristic of this robot is the existence of a closed form for its inverse kinematics which is valid within the task space. For the sake of brevity, the details of the closed form expression are omitted, but can be stated in a generic form for a robot with one degree of redundancy as

$$\mathbf{q} = \mathbf{g}(\mathbf{p}, u), \tag{1}$$

where $\mathbf{q}$ is a 7-dimensional vector denoting the joint configuration, $\mathbf{p}$ a 6-dimensional vector denoting the position and orientation of the end-effector, and u is the redundancy parameter. Differentiating (1) yields

$$\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\dot{\mathbf{p}} + \mathbf{N}(\mathbf{q})\dot{u}, \tag{2}$$

where $\mathbf{G}(\mathbf{q})$ is a 7×6 matrix and $\mathbf{N}(\mathbf{q})$ a 7-dimensional vector representing the null space of the manipulator Jacobian. The existence of the closed form solution is key for the developments in Sections 3 and 4.

The problem considered in this paper is for the end-effector to go from a given initial pose $\mathbf{p}_0$ at time t_0 to a given final pose $\mathbf{p}_f$ at time t_f via a known trajectory $\mathbf{p}(t)$ in Cartesian space. The initial joint configuration $\mathbf{q}_0$ is supposed to be known. The two criteria considered are the joint speed norm, $f_1 = 1/2\|\dot{\mathbf{q}}(t)\|^2$, and the aerodynamic interference cost function, $f_2 = f_2(\mathbf{q}(t))$.

III. Variational Formulation using the Weighting Method

The objective is to find the optimal joint trajectory that results in the minimum of the integral cost, J_w, of the linear combination of the two criteria [6] or find

$$J_w = \min_{\mathbf{q}(t)} \int_{t_0}^{t_f} f_1(\dot{\mathbf{q}}(t)) + w f_2(\mathbf{q}(t))dt, \tag{3}$$

subject to

$$\mathbf{f}(\mathbf{q}(t)) = \mathbf{p}(t), \tag{4}$$

and the boundary conditions

$$\mathbf{q}(t_0) = \mathbf{q}_0,\ \mathbf{p}(t_f) = \mathbf{p}_f, \tag{5}$$

where $\mathbf{f}$ is the forward kinematics mapping, and w the weighting factor.

It can be shown that the necessary Euler-Lagrange conditions for optimality yield a system of eight ordinary differential equations in $(\mathbf{q}, \lambda)$ [6], where λ is a scalar. These differential equations can be written analytically because of the existence of an analytical expression for the null space $\mathbf{N}(\mathbf{q})$ of the manipulator Jacobian. This improves the accuracy and speed of the computations. The added necessary conditions for optimality arising from the boundary conditions (5) are $\mathbf{q}(t_0) = \mathbf{q}_0$ and $\lambda(t_f) = 0$ [6]. At this point, it can be observed that the resulting two point Boundary Value Problem (BVP) reduces to a one dimensional search: find $\lambda(t_0)$ such that $\lambda(t_f) = 0$, which simplifies the process of obtaining all the stationary solutions to the variational problem.

We take advantage of the simple access to the stationary solutions for various values of the weight, $-10 \leq w \leq 20$, and calculate the corresponding values of the functionals $F_1 = \int_{t_0}^{t_f} f_1(\dot{\mathbf{q}}(t))dt$ and $F_2 = \int_{t_0}^{t_f} f_2(\mathbf{q}(t))dt$. Figure 2 depicts the variation of F_2 versus F_1 as the weight varies. The resulting curve is denoted by $\mathcal{C}$ and any point, $z = (F_1, F_2)$, on $\mathcal{C}$ is termed *objective vector.*

The discussion on the weighting method is primarily based on the following important geometric observation. For a given weight, there exists a stationary solution when the line with the slope $-1/w$ is tangent to $\mathcal{C}$. For example, when only F_1 is considered ($w = 0$), the vertical line is tangent to $\mathcal{C}$ at three different objective vectors z_1, z_2 and z_3 as seen in Figure 2. It can be noticed that z_1 is the global minimum, z_3 a local minimum and z_2 neither a minimum nor a maximum.

Numerical algorithms for calculus of variations and optimal control mostly rely on the first order necessary conditions and as a result, do not guarantee a minimum. For

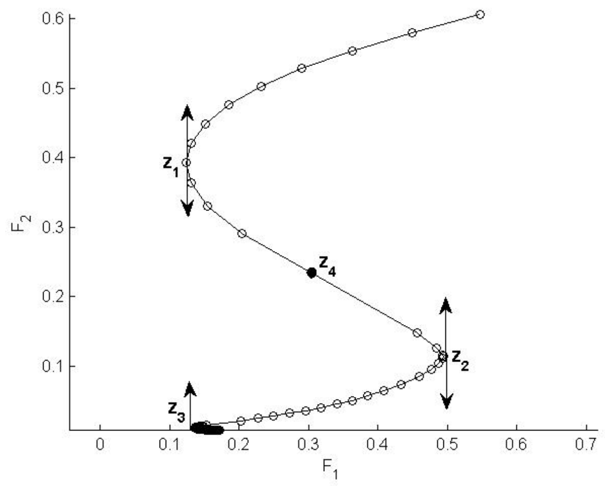

Fig. 2. Variation of F_2 versus F_1 as the weight w varies between -10 and 20.

example, for $w = 0$, z_2 could be a solution and increasing the weight has the opposite effect on F_2 as expected. The source of this problem is the failure of the numerical algorithm to capture a minimum and not the weighting method itself. However, if we assume that the numerical algorithm is able to generate a minimum, still two major problems can be identified:

- By changing the weights continuously, it is possible to jump from one part of $\mathcal{C}$ to another. For example, for $w = 0$, z_1 could be the optimal solution. By increasing the weight, the minimum (local) objective vector moves along $\mathcal{C}$ downwards until z_4 is reached. At z_4, although it is not quite obvious in Figure 2, the tangent starts to be above $\mathcal{C}$ and as a result, the corresponding objective vector is not a minimum. On the other hand, a unique minimum can be found near z_3.

- A uniform distribution of the weights does not necessarily result in a uniform distribution of the objective vectors. This can be seen in Figure 2 where the "o" are not equally spaced on $\mathcal{C}$. It is also possible that at some point on $\mathcal{C}$, small changes in weight result in drastic changes in objective vector (for example, between z_1 and z_4). On the other hand, it might be possible that for some points, even large changes in weight would not result in any noticeable changes in objective vector (for example, near z_3). The geometric justification is that the curvature of $\mathcal{C}$ is not constant, confirming the non Lipschitz nature of objective vectors as a function of weights, as stated in [7].

In light of the above discussion, we can see the difficulties of controlling the objective vector using the weighting method. The shortcomings of this method have also been reported in [4].

IV. Generation of Pareto Optimal Solutions with Dynamic Programming

In the first part of this section, we show how the same global minimum solution as obtained through variational calculus to the to the problem (3)-(5) can be generated with dynamic programming. In order to reduce the search space, joint limits and joint speed limits are introduced:

$$\mathbf{q_{min}} \leq \mathbf{q}(t) \leq \mathbf{q_{max}}, \tag{6}$$

$$\dot{\mathbf{q}}_{\mathbf{min}} \leq \dot{\mathbf{q}}(t) \leq \dot{\mathbf{q}}_{\mathbf{max}}. \tag{7}$$

However, only joint trajectories far from these limits will be considered for comparison purposes.

Dynamic programming has already been used to generate time optimal joint trajectories for nonredundant manipulators [11], [3] or for known joint paths [10]. We follow here the same approach, except that the end-effector path parameter is replaced by the redundancy parameter u as the state of the system. Substituting 1) and (2) in the formulation given by Equations (3)-(5) with the constraints (6) and (7) yields the following variational problem:

$$J_w = \min_{u(t)} \int_{t_0}^{t_f} \Phi_w(t, u(t), \dot{u}(t))dt, \tag{8}$$

subject to

$$u(t) \in \mathcal{A}(t), \tag{9}$$

$$\dot{u}(t) \in \mathcal{B}(t, u(t)), \tag{10}$$

and the boundary conditions

$$u(t_0) = u_0, u(t_f) \in \mathcal{A}(t_f), \tag{11}$$

where $\mathcal{A}(t)$ and $\mathcal{B}(t, u(t))$ represent, respectively, the admissible values of the redundancy parameter and its speed at any given time. Problem (8)-(11) might be solved by the numerical resolution of the Hamilton-Jacobi-Bellman (HJB) partial differential equation [3], or by stating its discretized version as a discrete dynamic programming problem [10], [11]. The discrete dynamic programming is preferred for its simplicity, which yields:

$$J_w^N(u_0) = \min_{\{u_i, i=0..N\}} \sum_{i=1}^{N} \Phi_w(i, u_i, \dot{u}_i)\tau, \tag{12}$$

subject to

$$u_i \in \mathcal{A}_i, \tag{13}$$

$$\dot{u}_i \in \mathcal{B}_i(u_i), \tag{14}$$

using the Euler approximation

$$\dot{u}_i = \frac{(u_i - u_{i-1})}{\tau}, \tag{15}$$

and the boundary conditions

$$u_N \in \mathcal{A}_N, \tag{16}$$

where τ is the discretization step time, $N = [\frac{t_f - t_0}{\tau}]$ is the number of steps, $u_i = u(i\tau)$, and $J_w^N(u_0)$ the minimum performance criterion at the step N for a joint trajectory (i.e. a sequence of u_k) starting from u_0. It is now possible to apply the Bellman optimality principle [2] to obtain:

$$J_w^k(u_0) = \min_{u_{k-1} \in \mathcal{A}'_{k-1}} \Phi_w(k, u_k, \dot{u}_k)\tau + J_w^{k-1}(u_0) \quad (17)$$

where $k = 1 \dots N$ and $\mathcal{A}'_{k-1} = \mathcal{A}_{k-1} \cap \{u_{k-1} \mid \frac{(u_i - u_{i-1})}{\tau} \in \mathcal{B}_k(u_k), u_k \in \mathcal{A}_k\}$. The dynamic programming Equation (17) could also be formulated through the common approach of using a return function, defined as the minimum performance criterion reaching the final state, but the two approaches result in the same solution. Finally, to solve Equation (17), we propose the following algorithm:

- Stage 1: build a grid in the (t, u) space. This grid can already embed (13) using the closed form solution to the inverse kinematics (1) in conjunction with the joint limits (6). Set the minimum performance criterion to infinity for each node $(k, u_{k,i})$.
- Stage 2: at step k, iterate over all $u_{k,i}$ satisfying Equation (13). For each $u_{k,i}$, find all the nodes $(k+1, u_{k+1,j})$ satisfying Equation (13) such that $\dot{u}_{k+1,j}$ satisfies Equation (14), with $\dot{u}_{k+1,j}$ being calculated with the Euler approximation Equation (15). Calculate the performance criterion and compare this performance criterion with the current minimum performance criterion. Replace the current minimum performance criterion if it is higher. Set the node $(k, u_{k,i})$ as the predecessor.
- Stage 3: Repeat until the step $N - 1$ is reached.
- Stage 4: Take the minimum of the minimum performance criterion at step N.

Euler integration does not allow the algorithm to reach the step N. This is the reason Stage 4 has been introduced. This could be simplified by adding an idle node at step $N + 1$ connected to all the nodes at step N without any constraints. Note that this algorithm moves forward, whereas if a return function had been used, the corresponding algorithm would have moved backwards. The validity of the proposed algorithm at the boundary of $\mathcal{A}(t)$ and $\mathcal{B}(t, u(t))$ has not been investigated. This is not a significant issue considering that only joint trajectories far from the limits are used for comparison purposes. For $w = 0$, Figure 3 illustrates the variation of the redundancy parameter as a function of the time for the three stationary solutions corresponding to z_1, z_2 and z_3 and the optimal solution obtained from the dynamic programming approach. It can be observed that there is a good agreement between the stationary solution corresponding to z_1, which is the global minimum, and the solution obtained from the dynamic programming approach. These results have been obtained with $N = 10$ for 8 seconds of run for a total of 515 nodes. Finally, Figure 3 shows that the stationary solution corresponding to z_3 does not satisfy the joint limits, because in Section 2, no constraints were included.

At this point, it can be concluded that the same global minimum solution to the problem given by Equations (3)-(5) can be obtained using Dynamic Programming. It is shown

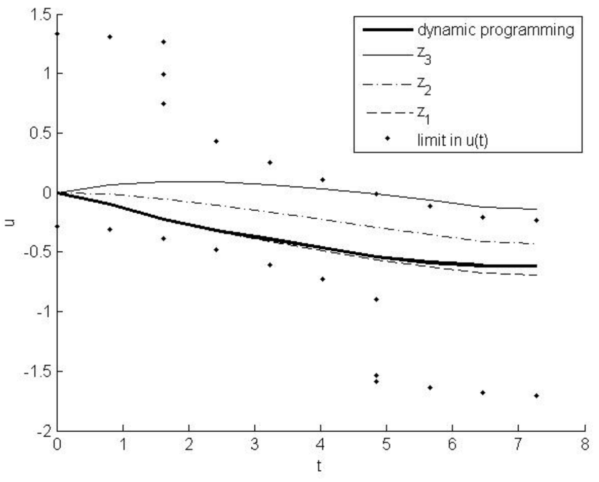

Fig. 3. $u(t)$ for Dynamic Programming and stationary solutions of the Euler-Lagrange equation.

now how the dynamic programming approach developed above can be modified to generate solutions which are optimal in a multiple criteria sense. These solutions are said to be Pareto optimal (also termed efficient or non dominated) with the definition given below [7].

Definition 4.1 (Pareto optimality): assume that n criteria with scalar values are to be minimized, an objective vector z^* is Pareto optimal if there does not exist another objective vector $z \in Z$ such that $z_i \leq z_i^*$ for all $i = 1 \dots n$ and $z_j < z_j^*$ for at least one index j.

Definition 4.1 introduces only the global Pareto optimality, which can be defined in the solution space as well. Another definition needs to be provided for local Pareto optimality.

Definition 4.2: an objective vector z^* is locally Pareto optimal if the corresponding solution is Pareto optimal only in a neighborhood.

Both local and global Pareto optimal solutions can form a set, because their number can be infinite. Figure 4 reproduces Figure 2 when any joint trajectory violating the joint limits or the joint speed limits is removed. The Pareto optimal and the locally Pareto optimal sets are determined and shown on the resulting curve.

From a mathematical point of view, every Pareto optimal solution is an equally acceptable solution. Hence, a multiple criteria optimization method might be evaluated by its ability to generate a better representation of the complete Pareto optimal set through a more uniform sampling. This was one the major weaknesses of the weighting method as explored in Section 3. To summarize, the weighting method lacks control of the objective vectors through the weights and is unable to generate solution in the nonconvex part of the Pareto optimal set [4] (it is impossible to generate objective vectors on the portion of the curve $\mathcal{C}$ between z_2 and z_4 in Figure 2). However, for a given set of nonnegative weights, the weighting method [7] guarantees Pareto optimal solutions, although these solutions are more likely to be only just local

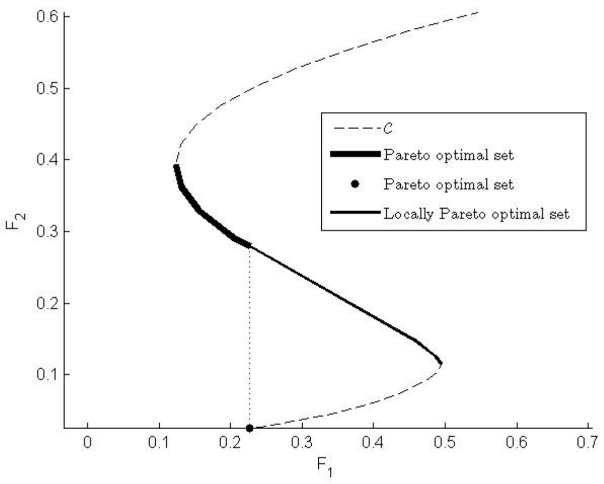

Fig. 4. Locally Pareto optimal set and Pareto optimal set.

Pareto optimal.

Let us consider again the dynamic programming approach presented above. The idea proposed is to embed directly the Pareto optimality or dominance concept within the algorithm used to generate the optimal solution (as a result, the dynamic programming equation Equation (17) is not the proper formulation anymore). Stages 1, 2 and 4 of this algorithm are modified as follows, while Stage 3 does not change.

- Stage $1^{'}$: build a grid in the (t, u) space. This grid can already embed (13) using the closed form solution to the inverse kinematics (1) in conjunction with the joint limits (6). Set the list of optimal objective vectors to void for each node $(k, u_{k,i})$.
- Stage $2^{'}$: at step k, iterate over all $u_{k,i}$ satisfying (13). For each u_i, find all the nodes $(k+1, u_{k+1,j})$ satisfying (13) such that $\dot{u}_{k+1,j}$ satisfies (14) ($\dot{u}_{k+1,j}$ being calculated with the Euler approximation (15)). Calculate then the objective vector $z(u_{k,i}, u_{k+1,j})$ and apply the following dominance rules:
  - if $z(u_{k,i}, u_{k+1,j})$ is dominated by any element in the list of optimal objective vectors, discard it. Otherwise, add it to the list and set the node $(k, u_{k,i})$ as the predecessor for this particular objective vector.
  - if $z(u_{k,i}, u_{k+1,j})$ dominates any element in the list of optimal objective vectors, discard this element.
  - Stage 3: repeat until the step $N-1$ is reached.
  - Stage $4^{'}$: apply the dominance rule for all the optimal solutions at step N.

We predict that this modified algorithm should be able to generate all the Pareto optimal solutions. Although very attractive, the algorithm might become computationally intractable. As we show below with a simple calculation, the number of nondominated objective vectors grows exponentially with the dimension the grid. Assume that at step k, each node can reach r_k nodes at step $k+1$. In the worst case scenario, where no nondominated objective vectors are discarded, the number of nondominated objective vectors at step $k+1$ is multiplied by r_k, which gives recursively a total of $\prod r_k$. This problem is more pronounced when the discretization in the (t, u) space becomes finer, and as the number of performance criteria increases. These are manifestations of the so-called *curse of dimensionality* [2].

It is possible to cope with this problem using heuristics whose main objective is to control the number of nondominated objective vectors at each step k. As an example, it is suggested to limit the number of nondominated objective vectors at each node using the following criterion: keep the $p, (p \geq 1)$ nondominated objective vectors with the smallest joint speed norm and discard the rest. This heuristic has been implemented with $p = 3$ and the results are displayed in Figure 5. The resulting nondominated objective vectors agree well with the Pareto optimal set and capture perfectly its non connectivity. This is the natural advantage of the dynamic programming approach that can offer a framework in which there is a great degree of freedom in choosing various heuristics. However, special attention should be given to the choice of such heuristics as they might result in losing Pareto optimality (global).

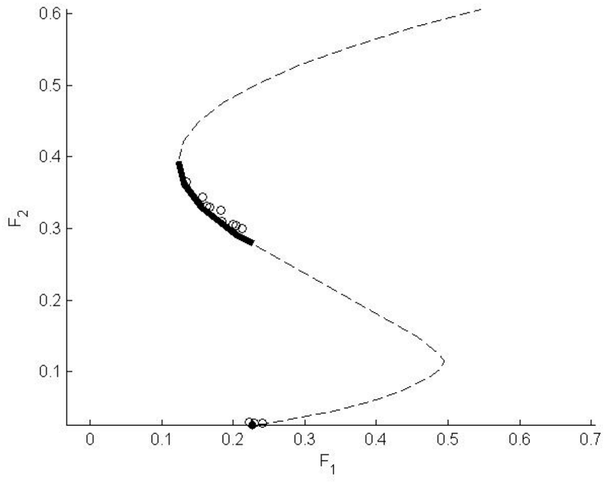

Fig. 5. Nondominated objective vectors generated by the modified dynamic programming approach.

V. CONCLUSIONS AND FUTURE WORK

A dynamic programming approach was proposed and applied to the model of a 7-DOF redundant manipulator in order to find the optimal joint trajectories considering joint speed norm and the aerodynamic interference as the two performance criteria. This approach provides better means to identify the Pareto optimal set than the traditional weighting method which avoids addressing multiple criteria problems through transforming them into a single criterion. Various simple or complex heuristics can be conveniently implemented at each node of the grid to reduce the computational load of the search and eliminate certain objective vectors based on a preference, as confirmed by simulation. The handling of constraints within the dynamic programming approach still needs to be further investigated. Additional criteria and constraints such as operational constraints of the CTS system and collisions will be considered as the immediate extensions of this work.

REFERENCES

[1] M. Ahmadi, D. Orchard, F. C. Tang, ”Captive Trajectory Simulation: On Robotic Performance Effects”, *in Mechatronics and Robotics Conference*, Aachen, Germany, 2004, pp. 802-807.

[2] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[3] A.J. Cahill and M.R. James and J.C. Kieffer and D. Williamson, ”Remarks on the Application of Dynamic Programming to the optimal Path Timing of Robot Manipulators”, *International Journal of Robust and Nonlinear Control*, vol. 8, 1998, pp. 463-482.

[4] I. Das and J.E. Dennis, ”A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems”, *Structural Optimization*, vol. 14, 1997, pp. 63-69.

[5] M. Galicki, ”Planning of Robotic optimal Motions in the Presence of Obstacles”, *The International Journal of Robotics Research*, vol. 17, no. 3, 1998, pp. 248-259.

[6] D.P. Martin and J. Baillieul and J.M. Hollerbach, ”Resolution of Kinematic Redundancy Using Optimization Techniques”, *IEEE Transactions on Robotics and Automation*, vol. 5, no. 4, 1989, pp. 529-533.

[7] K. M. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, 1999.

[8] Y. Nakamura and H. Hanafusa, ”optimal Redundancy Control of Redundant Manipulators”, *The International Journal of Robotics Research*, vol. 6, no. 1, 1987, pp. 32-42.

[9] Y. Shen and K. Huper, ”optimal Trajectory Planning of Manipulators Subject to Motion Constraints”, *IEEE Transactions on Robotics and Automation*, 2005, pp. 9-16.

[10] Shin, K. G. and N. D. McKay, ”A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators”, *IEEE Transactions on Automatic Control*, vol. AC-31, no. 6, 1986, pp. 491-500.

[11] Singh, S. and M. C. Leu, ”optimal Trajectory Generation for Robotic Manipulators using Dynamic Programming”, *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 109, 1987, pp. 88-96.