2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

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 studies the problem of generating optimal joint trajectories for redundant manipulators when multiple criteria need to be considered and proposes a novel approach based on Dynamic Programming and the use of the Pareto optimality condition. The drawbacks of the traditional weighting method in optimization for generating the Pareto optimal set are discussed and an alternate approach using dynamic programming is proposed. 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 particular criteria. The results illustrate that the dynamic programming approach provides a better approximation of the Pareto optimal set and a 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, which is generally done by optimizing a performance criterion. The optimization problems arising from redundancy resolution involve functions of robot variables such as joint configurations, joint speeds, 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 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 and sophisticated numerical algorithms. Recently, an optimal control formulation [5] has been proposed, which uses the joint torques as the control inputs and the joint torque limits, end-effector path, and workspace obstacles as constraints. This problem is then solved by using the negative formulation of Pontryagin's Maximum Principle. A variational formulation has been proposed in [9] with kinematic compliant constraints and is solved via Newton iterations on a discretized Lagrange function. In these works, only a single criterion or a linear combination of multiple criteria is considered.

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

In complex applications such as the one presented in this paper, naturally come several criteria that are to be considered for optimization concurrently. The most common method to solve multiple criteria problem is the weighting method [6], [9], [10], [11], which consists of linearly combining the criteria with some weights, hence transforming the original problem into a single criterion problem. However, this approach suffers from several major drawbacks [4]. In particular, it is difficult to predict the variation of the criteria for the optimal solution as the weights vary, which might sometimes show the opposite variation as expected. This problem is illustrated for a specific task with two criteria to be executed by the model of the 7-DOF redundant manipulator which is part of a Captive Trajectory Simulation (CTS) system installed inside a supersonic wind tunnel, as described in Section 2.

In this paper, we propose to use Dynamic Programming [2] to the manipulator redundancy resolution problem with multiple criteria. By implementing the optimality in a multiple criteria sense, i.e. using the Pareto optimality condition, at each stage of the dynamic programming process, it is possible to avoid the use of the weighting method and its associated drawbacks. Pareto optimality refers to the condition where it is not possible to find a direction that all criteria can be improved concurrently. This new approach has been successfully implemented on the two-criteria task, as mentioned above, resulting in a good sampling of the Pareto optimal set for the 7-DOF manipulator.

This paper is organized as follows. Section 2 describes the Captive Trajectory Simulation (CTS) system and the redundant manipulator involved in this system as well as the task and the two criteria used for this study. In section 3, a variational formulation with linearly combined criteria for redundancy resolution is used. The important drawbacks associated with the weighting method are also highlighted. Section 4 starts by proposing a dynamic programming approach with linearly combined criteria for redundancy resolution and shows how the results compare to those of the variational formulation. Then this approach is expanded by implementing the Pareto optimality condition at each stage of the dynamic programming process and the results are presented. Section 5 concludes the findings and proposes future work.

1-4244-0602-1/07/$20.00 ©2007 IEEE.

II. PROBLEM DEFINITION

The 7-DOF manipulator used in this paper is modeled after a Captive Trajectory Simulation (CTS) system with the objective of emulating the store (any object released from an aircraft) trajectory. This manipulator, operating within a supersonic wind-tunnel as illustrated in Fig. 1, holds, as its end-effector, the model of a store mounted on a sensitive 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 model. The role of the robot is to provide the desired position and orientation for the store with respect to the parent aircraft model. However, the presence of the manipulator in this area is undesirable as it disturbs the flow properties around the aircraft and can also impose additional disturbing loads on the manipulator links. Therefore, it is desirable to have the main body of the manipulator 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 manipulator operates underneath the wing of the airplane model. This constitutes one of the two criteria used in this study. The other criterion is the commonly used joint speed norm. Further details about the CTS systen in wind-tunnel can be found in [1].

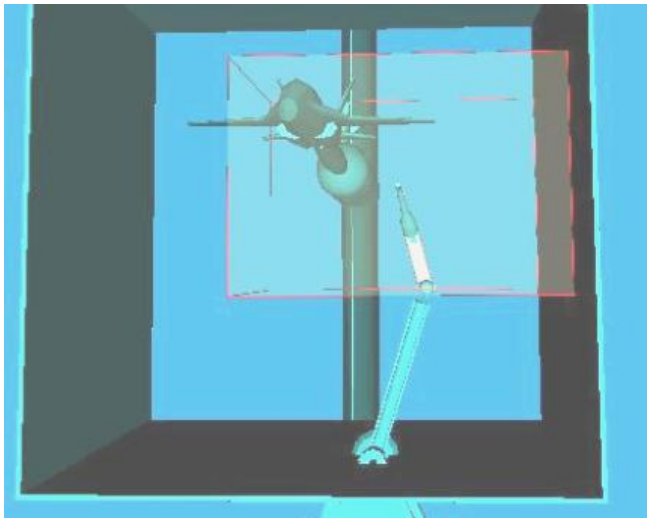

Fig. 1. The CTS manipulator inside the wind tunnel.

The manipulator has a box-shaped 6-DOF task space as illustrated in Fig. 1. An attractive characteristic of this manipulator 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 manipulator 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 smooth known trajectory $\mathbf{p}(t)$ in Cartesian space. The initial joint configuration $\mathbf{q}_0$ is supposed to be known. Mathematically, the two criteria considered can expressed as $f_1 = 1/2\|\dot{\mathbf{q}}(t)\|^2$ for the joint speed norm and $f_2 = f_2(\mathbf{q}(t))$ for the aerodynamic interference cost function.

III. VARIATIONAL FORMULATION WITH THE WEIGHTING METHOD

The problem described above can be formulated as a problem in calculus of variation where criteria are linearly combined to form a single criterion [6]. With this framework, the objective is to find the optimal joint trajectory $\mathbf{q}(t)$ that minimizes the integral cost J_w 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 the manipulator kinematic constraint

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

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 weight associated with the second criterion.

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 for our manipulator 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 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 makes easy the process of obtaining all the stationary solutions, i.e. finding solutions satisfying the Euler-Lagrange conditions for optimality.

To analyze the behavior of the weighting method, we propose to plot the values of the functionals $F_2 = \int_{t_0}^{t_f} f_2(\mathbf{q}(t))dt$ versus $F_1 = \int_{t_0}^{t_f} f_1(\dot{\mathbf{q}}(t))dt$ for the stationary solutions as the weight varies. Note that there

might be more than one stationary solution for a given weight as illustrated for $w = 0$ below. Fig. 2 presents the resulting curve denoted by $\mathcal{C}$. Any point $z = (F_1, F_2)$ on $\mathcal{C}$ is termed an *objective vector*. The objective vectors "o" in Fig. 2 have been obtained for a uniform distribution of the weight over the interval $[-10, 20]$.

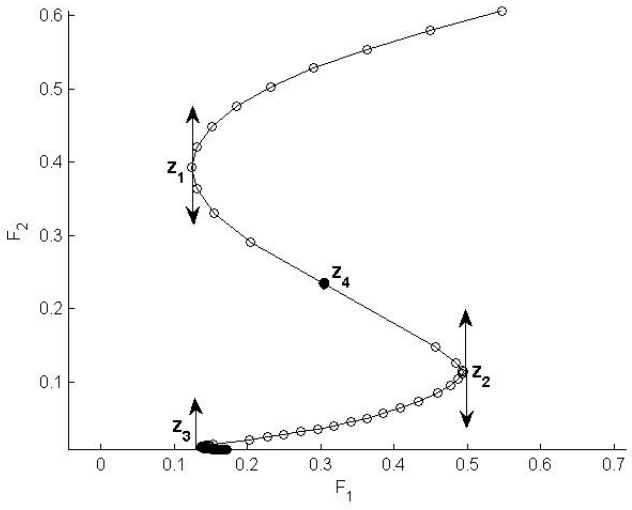

Fig. 2. F_2 versus F_1 for the stationary solutions as the weight w varies between -10 and 20.

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 the first criterion is considered, i.e. when $w = 0$, the vertical line is tangent to $\mathcal{C}$ at three different objective vectors z_1, z_2 and z_3 as seen in Fig. 2. It can be noticed that z_1 corresponds to the global minimum and z_3 to a local minimum while z_2 corresponds neither to a minimum nor a maximum.

Numerical algorithms for calculus of variations and optimal control mostly rely on the first order necessary conditions (respectively Euler-Lagrange and Pontryagin's Maximum Principle) and as a result, do not guarantee a minimum. For 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 Fig. 2, the tangent starts to lie 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 weight does not necessarily result in a uniform distribution of the objective vectors. This can be seen in Fig. 2 where the objective vectors "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 of this fact 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. These difficulties have already been reported in the multiple criteria research community [4].

IV. IMPLEMENTING THE PARETO OPTIMALITY CONDITION WITH DYNAMIC PROGRAMMING

In the first part of this section, we show how the same global minimum to the variational problem (3)-(5) (corresponding to z_1) can be obtained using a dynamic programming approach. 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. The closed-form inverse kinematics (1) and its differential form (2) can be substituted in (3)-(5) as well as in the constraints (6)-(7) to get the equivalent one-dimensional 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

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

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

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 derivative at time t. Problem (8)-(11) might be solved by 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 latter approach is preferred for its simplicity, which yields:

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

subject to

$$u_i \in \mathcal{A}_i, \qquad (13)$$

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

using the Euler approximation for the derivative:

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

and the boundary conditions

$$u_N \in \mathcal{A}_N, \qquad (16)$$

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 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:

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

where $k = 1 \ldots N$ 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 in this case, the two approaches result in the same solution. Finally, to solve (17), we propose the following algorithm:

- Stage 1: build a grid in the (t, u) space. Calculate $\mathbf{q}$ at each node of the grid with the closed-form inverse kinematics (1). Discard the node if the resulting $\mathbf{q}$ does not satisfy the joint limits (6). Set the minimum performance criterion to infinity for each node $(k, u_{k,i})$ left.
- Stage 2: at step k, iterate over all $u_{k,i}$. For each $u_{k,i}$, find all the nodes $(k+1, u_{k+1,j})$ such that $\dot{u}_{k+1,j}$ satisfies (14), with $\dot{u}_{k+1,j}$ being calculated with the Euler approximation (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 and set the node $(k, u_{k,i})$ as the predecessor.
- Stage 3: Repeat until step $N-1$ is reached.
- Stage 4: Take the minimum of the minimum performance criterion at step N.

Stage 4 has been introduced because Euler integration (15) does not allow the algorithm to reach step N. 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 the proposed algorithm moves forward, whereas if a return function had been used, the corresponding algorithm would have moved backwards. The validity of this 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 constraints are used for comparison purposes. For $w = 0$, Fig. 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

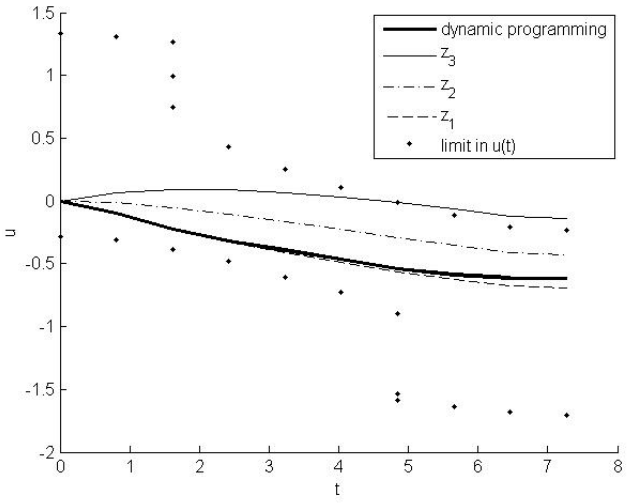

Fig. 3. Comparison of $u(t)$ between the dynamic programming approach and the stationary solutions of the variational formulation for $w = 0$.

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 8 seconds and $N = 10$ for a total grid size of 515 nodes. Note that Fig. 3 shows that the stationary solution corresponding to z_3 does not satisfy the joint limits: these constraints were not included in the variational formulation in Section 2.

We have illustrated that the same global minimum to the variational problem (3)-(5) can be retrieved using a dynamic programming approach. It will be shown now how this approach can be modified to directly generate solutions which are optimal in a multiple criteria sense without resorting to the weighting method. Such solutions are said to be Pareto optimal (also termed efficient or non dominated) according to the following definition.

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 $i = 1 \ldots n$ and there is least one index j such that $z_j < z_j^*$ [7].

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 [7].

The number of both locally and globally Pareto optimal solutions is generally infinite and as such, these solutions form sets. Fig. 4 reproduces Fig. 2 when any joint trajectory violating the joint limits or the joint speed limits has been removed. The Pareto optimal and the locally Pareto optimal sets are determined and shown on the resulting curve.

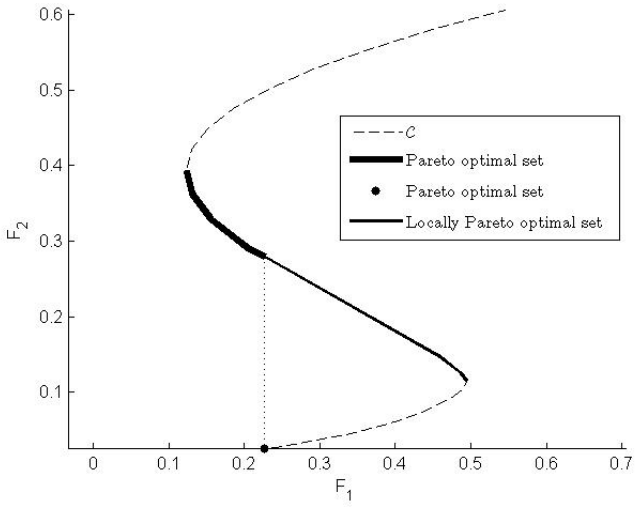

Fig. 4. Illustration of the locally Pareto optimal set and the Pareto optimal set.

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 of the major weaknesses of the weighting method as explored in Section 3. To summarize, we illustrated that 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 Fig. 2). However, at least, 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 locally Pareto optimal.

Consider, again, the dynamic programming approach presented above. We propose to directly implement the Pareto optimality condition or the dominance concept within the algorithm used to generate the optimal solution (as a result, (17) is not the proper dynamic programming equation 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. Calculate $\mathbf{q}$ at each node of the grid with the closed-form inverse kinematics (1). Discard the node if the resulting $\mathbf{q}$ does not satisfy 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 $u_{k,i}$. For each $u_{k,i}$, find all the nodes $(k+1, u_{k+1,j})$ such that $\dot{u}_{k+1,j}$ satisfies (14), with $\dot{u}_{k+1,j}$ being calculated with the Euler approximation (15). For each pair of nodes $((k, u_{k,i}), (k+1, u_{k+1,j}))$, iterate over the list of optimal objective vectors at the node $(k, u_{k,i})$ and calculate the corresponding objective vector for the node $(k+1, u_{k+1,j})$.
  - if this objective vector is dominated by any element in the current list of optimal objective vectors at the node $(k+1, u_{k+1,j})$, discard it. Otherwise, add it to the list and set the node $(k, u_{k,i})$ as the predecessor.
  - if this objective vector dominates any element in the list of optimal objective vectors at the node $(k+1, u_{k+1,j})$, 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.

However, as we show below, with a simple calculation, the number of nondominated objective vectors might grow exponentially with the dimension the grid. Indeed, 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 vector is discarded, the total 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].

This problem can be addressed using a heuristic at Stage $2^{'}$ whose objective is to bound the number of nondominated objective vectors generated at each step k. As an example, the number of nondominated objective vectors at each node of the grid could be limited by keeping only p nondominated objective vectors with the smallest value for F_1 ($p \geq 1$) and discarding the rest. This heuristic has been implemented with $p = 3$ and the results are displayed in Fig. 5. It is shown in Fig. 5 that the set of objective vectors generated by the modified dynamic programming approach agree well with the Pareto optimal set and, more importantly, captures its non connectivity. The freedom in choosing a heuristic is very large. However, it remains to be investigated whether the use of a given heuristic guarantees global or at least local Pareto optimality.

While we think that our approach, based on dynamic programming, provides a good means of estimating the Pareto optimal set, the final selection of one solution is still

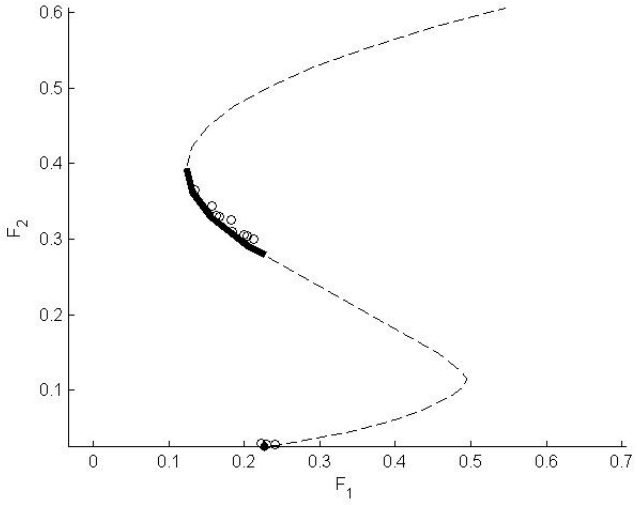

Fig. 5. Nondominated objective vectors generated by the modified dynamic programming approach.

required. This paper does not address this question, but two main approaches could be suggested. The first method would be to define "secondary criteria" (the "primary criteria" being defined as the ones used in the optimization process) that can be applied with a simple or complex preference that suits a particular application. Such decision making could even be done by an operator through the visualization of the solutions. With this approach, any criterion critical to the application should be included in the set of "primary criteria". The second method would be to make the final selection based on certain characteristics of the Pareto optimal set which is only possible to do after the set has been identified. For example, one may choose the median of the Pareto optimal solutions, the closest Pareto optimal solution to the average, or use any other criteria applied to the set.

V. CONCLUSIONS AND FUTURE WORKS

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 the joint speed norm and the aerodynamic interference as the two performance criteria. This approach, which avoids transforming the original problem into a single criterion problem, provides better means to identify the Pareto optimal set than the traditional weighting method. Various simple or complex heuristics can be conveniently implemented at each node of the grid to reduce the computational load of the search and to eliminate certain objective vectors based on a preference. The optimality behavior of the proposed approach at the boundary of the constraints needs to be further investigated. Additional criteria and constraints such as the operational constraints of the CTS system and the collision avoidance will be considered as immediate extensions of this work.

REFERENCES

[1] M. Ahmadi, D. Orchard and 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, M.R. James, 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, 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] K.G. Shin 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] S. Singh 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.