On Computing the Global Time-Optimal Motions of Robotic Manipulators in the Presence of Obstacles

Zvi Shiller, Member, IEEE, and Steven Dubowsky, Member, IEEE

Abstract—A method for computing the time-optimal motions of robotic manipulators is presented that considers the nonlinear manipulator dynamics, actuator constraints, joint limits, and obstacles. Using a previously developed method for computing the time-optimal motions along specified paths, the optimization problem is reduced to a search for the time-optimal path in the n-dimensional position space. A small set of near-optimal paths are first efficiently selected from a grid, using a branch and bound search and a series of lower bound estimates on the traveling time along a given path. These paths are further optimized with a local path optimization to yield the global optimal solution. Obstacles are considered by eliminating the collision points from the tessellated space and by adding a penalty function to the motion time in the local optimization. The computational efficiency of the method stems from the reduced dimensionality of the searched space and from combining the grid search with a local optimization. The method is demonstrated in several examples for two- and six-degree-of-freedom (DOF) manipulators with obstacles.

I. INTRODUCTION

The potential increase in the productivity and efficiency of industrial robots over human operators is often the major reason for their use in many industrial applications. The productivity of robotic systems can be maximized by planning robot motions to be time optimal. Optimal motions can yield reduced cycle times, increased system utilization, and thus improve the cost effectiveness of typical automated manufacturing systems.

The problem of optimal motion planning concerns the determination of robot motions that move the end-effector at a minimum time between two given positions and orientations while satisfying the limits on the actuator efforts and avoiding collision with obstacles. The exact solution of the time-optimization problem is found to be computationally difficult mainly due to the typically highly coupled and nonlinear robot dynamics. Using the optimality condition stated by the Pontryagin maximum principle, the optimization problem can be transformed to a 4n-dimensional two-point boundary value problem (TPBVP) for an n-degree-of-freedom (DOF) manipulator. The typically poor convergence of numerical methods for solving high-dimensional nonlinear TPBVP's makes this approach impractical for realistic manipulators with three or more DOF's. The presence of obstacles makes this approach even more difficult because of the complex inequality constraints imposed by the obstacles on the state space. This approach has been demonstrated only for simple two-dimensional systems with no obstacles, using a shooting method [4] or a more efficient gradient search [10] for solving the TPBVP.

A dynamic programming search over the entire 2n-dimensional state space can potentially yield the global optimal trajectory among obstacles [3], [13]. A grid search, however, suffers from what is known as the "dimensionality curse," and is therefore computationally impractical except for the simplest systems and for a low grid resolution. In fact, the solutions obtained in [13] for a two-link manipulator are far from optimal due to the low grid resolution used. More refined grids result in extensive computation times, rendering this approach impractical for obtaining reasonably near-optimal solutions. Based on the assumption that the time-optimal control is bang-bang, the optimal control problem was reduced to a parameter optimization by iterating on the switching times [7], [11]. This approach fails at singular arcs along which the time-optimal control is not necessarily bang-bang. Replacing singular arcs with multiple switches is possible but undesirable due to the large number of switches required (infinite to be exact) and the inaccuracies associated with every switch. Other attempts to solve this problem by simplifying the manipulator model resulted in nonoptimal solutions [6], [14], [22].

One approach to reduce the complexity of the problem and to facilitate a practical realization of time-optimal motion planning is to represent robot motions by a path and a velocity profile along the path. The path is then defined by the n-dimensional projection of the state space trajectory on the position space, and the velocity profile along the path is represented in a two-dimensional phase plane. This separation between the path and the velocity profile allows reducing the optimization problem to two smaller problems: a) computing the optimal velocity profile along a given path, and b) searching for the optimal path in the n-dimensional position space. The time-optimal velocity profile along a specified path can be computed with an algorithm that was originally developed by Bobrow et al. [1] and Shin and McKay [21], and was later modified to consider singular points and arcs.
The local optimal path can be obtained with a parameter optimization that iterates on path parameters, such as the control points of a B spline [18]. This approach is currently the only one to have been demonstrated computationally practical for realistic six-DOF manipulators.

With the exception of the dynamic programming search in the state space [3], [13], most of the methods developed to date obtain only local optimal motions that depend on the initial guess. Manually selecting an appropriate initial guess may prove to be difficult and time consuming, especially in the presence of obstacles, since a small number of obstacles may create a large number of potential local minima.

This paper presents a practical method for computing the global time-optimal motions of general robotic manipulators operating in cluttered environments that combines a grid search in the position space with a continuous local optimization. The optimization method is reduced to selecting the best path out of a large (but finite) number of paths represented by a tessellated position space, using a branch and bound search and a series of lower bound estimates on the traveling time along each path. This path is used as the initial guess for the local optimization to yield the global optimal solution. To guarantee global optimality even for low grid resolution, a number of best paths are selected for the local optimization.

The number of these paths depends on the grid size and the sensitivity of the cost function to path variations near the global optimum. Obstacles and joint limits are considered in the grid search by excluding grid points that are out of reach for the manipulator end-effector. The local optimization uses a penalty function to ensure that the optimal path does not approach obstacles or manipulator joint limits.

While it is virtually impossible to rigorously guarantee the global optimality of any solution to this problem without an exhaustive search, it is practically feasible to compute the global optimal path using this method, assuming that the desired optimal path is smooth and that the region of convergence around the optimal path is sufficiently large compared to the grid size. It is also assumed that small deviations in path geometry yield small deviations in the motion time along the path. Experience shows this assumption to be valid for the class of manipulators considered in this research. Solutions that do not satisfy these assumptions are generally undesirable because of their high sensitivity to such factors as model errors and disturbances.

Formulating the problem in the \( n \)-dimensional position space yields a significant (exponential) computational gain compared to a search in the \( 2n \)-dimensional state space. The presence of obstacles makes this approach even more efficient as it reduces the searched space and the number of initial paths to be considered. The use of a local optimization in conjunction with the grid search further enhances the efficiency of this approach by relaxing the demands on the grid resolution. The method, implemented on a microcomputer, \( \mu V A X \ II \), is demonstrated by several examples for planar two and spatial six-dimensional manipulators operating in two- and three-dimensional spaces with obstacles.

This paper is organized as follows: Section II describes the global search for a set of near-optimal paths. This includes a special grid representation of the workspace, the branch and bound search, and the lower bound approximations of the cost function. The local optimization is briefly discussed in Section III, and a summary of the implemented algorithm is provided in Section IV. Examples for two- and six-DOF manipulators with and without obstacles are presented in Section V.

II. Global Search in a Tessellated Space

The global search is aimed at selecting the initial guesses for the local optimization that are likely to converge to the global optimal path. Hypothetically, it may be possible to construct all possible paths along the grid between the end points and test each one for its optimal motion time. The one with the lowest time is obviously the global optimal for the given grid. It is, however, more efficient to generate the paths with lower bound estimates on the optimal time and use a branch and bound search to consider only a subset of all possible paths. The reduction of the search to the position space drastically reduces computation time because of the typical exponential complexity of grid search methods. The need to evaluate a large number of paths is not as costly since the computation is linear in the number of paths, and only a small subset of all possible paths are considered.

A. Work Space Representation

The workspace is represented by a uniform grid to reduce the number of feasible paths between the end points to a final set, and to facilitate a combinatorial search for the set of best paths. A typical point on the grid can represent any appropriate set of coordinates, such as the position and orientation of the end-effector \( X \) or a vector of the joint angles \( \theta \). To avoid loops and to eliminate paths with sharp turns, such as shown in Fig. 1, the position grid is augmented with a direction state to make the direction of departure from a grid point be a function of the direction of arrival to that point. Eliminating paths with loops and sharp turns contributes to the computational efficiency of this approach by drastically reducing the number of possible paths along the grid. It does not affect the ability of this approach to find the optimal path since sharp changes in the direction of motion may be feasible at high speeds due to the actuator constraints.

A typical grid point \( x_j \) is connected to its adjacent neighbors \( x_j' \), defined as

\[
x_j' = x_j + \Lambda_j d, \quad j = 1, \ldots, 3^a - 1
\]

where \( d \in \mathbb{R}^a \) is a vector of the typical grid sizes along the \( n \) axes, and \( \Lambda_j \in \mathbb{R}^{a \times a} \) is a diagonal matrix with the elements \( -1, 0, \text{ or } 1 \). The total number of distinct matrices \( \Lambda_j \) is \( 3^a \), and the total number of adjacent points is \( 3^a - 1 \), excluding the case when \( \Lambda = 0 \). The set of adjacent points \( x_j' \) represents a hypercube centered around point \( x_j \). A typical grid point in a two-dimensional space has \( 3^2 - 1 = 8 \) adjacent grid points, all shown in Fig. 2. In three dimensions, the number of adjacent points is \( 3^3 - 1 = 26 \).

Augmenting the position space with a direction state allows representing the search space by a directed graph with nodes \( \{ a_{i,j} \} \) and edges \( \{ e_{i,j} \} \). A typical node \( a_{i,j} \in x_j, \ j = \)
1, \(3^{n-1}\) represents grid point \(x_i\) and the \(j\)th direction of arrival to that point. A typical edge \(e_{i,k}\) connects a node belonging to grid point \(x_i\) to its neighbor in the \(k\)th direction. Each node has only one incoming edge and as many departing edges as desired. The departure angle is defined as the angle between the incoming and the departing edges. In Fig. 3, the incoming edge \(e_{1,2}\) from grid point 1 is connected to grid point 5. This node is connected to only three departing edges, in directions 1, 2, and 3. These directions were chosen to restrict the departure angle to 45°.

The complexity of a graph search is proportional to the number of edges. Here, the total number of nodes is \((3^n - 1)m^n\), where \(n\) is the number of states, and \(m\) is the number of points defined along each state (assuming the same tessellation for all states). The total number of edges is, therefore, \(q(3^n - 1)m^n\), where \(q\) is the number of departing edges from each node. This is exponentially better than the common state space representation. A state space grid for the same problem would have \(m^{2n}\) nodes and \((3^n - 1)m^{2n}\) edges. Dividing the number of nodes of the state space grid by the number of nodes of the augmented position space grid yields an exponential improvement of \(m^n/q\) in the complexity of the search algorithm.

Eliminating paths with sharp turns drastically reduces the total number of feasible paths between the end points. For example, in a 5 x 5 grid with three neighbors to each node, the total number of possible paths is reduced from 321 to 27. The 21 paths span the same space as the 321 ones and are more likely to represent the near-optimal path. The gains for higher resolution grids are even more substantial [15].

The approach presented in this paper is described for a Cartesian space implementation; however, the method is applicable to paths in joint space as well.

**B. Obstacle Representation**

To allow a search for paths in the Cartesian space, we define obstacle shadows as regions formed by grid points that are not accessible to the manipulator tip due to the presence of obstacles. Fig. 4 shows an obstacle shadow for one obstacle in the environment of a two-link planar manipulator. Any path that does not pass through an obstacle shadow is obstacle free. The obstacle shadow is generated by either testing each grid point for collision with an obstacle or by analytically computing its boundaries [5].

Obstacle shadows can be represented as a mapping of the configuration space obstacles to the Cartesian space. Define \(A\) as the set of all reachable points in the Cartesian space, mapped from the joint space \(B\) by the single valued forward kinematics function \(FK(\cdot)\):

\[
A = \{x | x = FK(y), y \in B\}.
\]

The inverse mapping from \(B\) to \(A\) for nonredundant manipulators is generally not unique due to the multiple solutions to the inverse kinematics problem [2]. To resolve this ambiguity, it is convenient to subdivide the joint space \(B\) to subsets \(B_i\) consisting each of points for which a single valued inverse kinematic solution exists. For two different points \(y, z \in B\) and \(y \in B_i\)

\[
FK(y) = FK(z) \quad \text{only if} \quad z \notin B_i.
\]

The intersections between two adjacent subsets constitute a set of singularity points at which the manipulator switches between different kinematic configurations. If \(B_i\) maps to \(A_i\) through (2), then the kinematic map between \(B_i\) and \(A_i\) is invertible.

A configuration space obstacle \(CO(b)\) due to obstacle \(b\) is defined as [9]:

\[
CO(b) = \{y | R(y) \cap b \neq \emptyset\}
\]

where \(R(y)\) represents the set of points occupied by robot links at joint positions \(y \in B\). We define the \(i\)th obstacle shadow \(OS(b)\) due to obstacle \(b\) as

\[
OS(b)_i = \{x | x = FK(y), y \in CO(b) \cap B_i\}.
\]

Only one of the obstacle shadows defined in (5) is considered in generating the grid. This may exclude obstacle-free paths that pass through kinematic singularity points; however, if...
such paths are the only admissible ones, then a search in the joint space is recommended.

C. Branch and Bound Search

We use a branch and bound search to select a set of best paths from all possible ones between the end points. This search consists of dividing the solution space into smaller subsets for which a lower bound on the objective function can be computed, using a simpler approximation of the problem. The lower bounds are used to branch the search toward the most promising subsets and to discard certain subsets from further consideration. The search is terminated when each subset has been shown to contain no better solution than the one already at hand. The best solution found during this search is a global optimum [12].

Here, the solution space consists of a finite set of all feasible paths between the given end points, a subset consists of one or more paths, and the objective function is the traveling time between the end points. Key to this approach is the selection of appropriate approximations of the cost function that are guaranteed to produce lower bounds on the traveling time along a given path or a set of paths. Several such approximations (called tests) are presented that are based on the physical and dynamic characteristics of the manipulator and its time-optimal paths. The most conservative but efficient approximations are used first, when the number of path candidates is large, and the more accurate but computationally expensive are used last.

The branch and bound search is demonstrated schematically in Fig. 5 for a simple example (see also Appendix A for a formal procedure). The node $P_s$ represents the set of all feasible paths. The best path from that set is selected using the first test $t_1$ as the cost criterion. This divides the space to two subsets, one consisting of the best path, represented by node $P_{s1}$, and the other consisting of all the remaining paths, represented by node $P_{s}$. Generally, node $P_{s}$ represents the remaining paths after $k$ best paths were excluded from the initial set. The cost obtained by the first test $t_1$ is the lowest lower bound on the traveling time along the path. It is also a lower bound on the subset $P_{s}$ from which this path was excluded since that subset contains no better paths. There are four tests, marked $t_1$ to $t_4$, that produce successively lower bounds on the objective function. Node $P_{s,k}$ represents the cost obtained by the $k$th test for the $k$th path. The last test is the exact solution, which is the optimal traveling time along this path. Based on this test, a decision whether to consider the path as a near-minimum can be made.

Submitting the first path to the next test $t_2$ produces the node $P_{s,1}$ with a cost of 3. The search proceeds by exploring the node with the lowest cost, in this case subset $P_s$ with a cost of 2. The best path in this subset, which is also the second best path in the initial set, is marked by the node $P_{s,2}$ with a cost of 2.5. Subsequent paths can have either the same or higher cost than the one already at hand. The search proceeds by either selecting more paths from the current subset of paths or by testing the existing paths with higher tests. At some point, the first path reaches a solution with a cost of 6. This establishes the upper bound for the optimal solution, which is updated every time a better solution is reached. Here, the search is terminated at node $P_{s}$ since it does not contain any path with a better time than the ones already selected. After the search is terminated, the optimal path is selected from all paths marked as solutions. In the case shown, the global optimal solution is 5.5. The proof of the global optimality of such a search is given in Appendix B.

In the example shown in Fig. 5, only five paths from the initial set were explored. All five were evaluated by the first test, three by the second test, and only two by the third and fourth tests. The decrease in the number of paths computed by the more accurate tests results in the computational efficiency of this approach. Clearly, the computational efficiency depends on the selection of the lower bound tests, although this may vary with the system and the objective function. For example, distance may provide an adequate measure for the motion time of Cartesian manipulators. For more complex systems, such as articulated manipulators with highly nonlinear and coupled dynamics, distance may be used only for the first crude approximation of the motion time, followed by more accurate tests that take into account system dynamics and path characteristics.

D. Time-Optimal Motions Along Specified Paths

The optimal motion time along the path represents the exact cost function for the global search. The lower bound estimates used for the branch and bound search are derived from the optimization method used to compute the optimal motion time [1], [20].

The method considers manipulator dynamics that can be
written in the form

$$M \ddot{\theta} + \dot{\theta}^T C \dot{\theta} + G = T$$

(6)

where $M$ is an $n \times n$ inertia matrix, $n$ is the number of the manipulator degrees of freedom, $C$ is an $n \times n \times n$ array of the coefficients of the centrifugal and Coriolis forces, $G$ is a vector of the gravity forces, $T$ is the vector of actuator efforts, and $\theta$, $\dot{\theta}$, and $\ddot{\theta}$ are the joint displacements, velocities, and accelerations, respectively. Using the kinematic constraints along the path

$$g(\theta, S) = 0; \quad g \in \mathbb{R}^{n-1}$$

(7)

one can reduce the multistate optimization to a two-state problem, the two states being the position $S$ and the velocity $\dot{S}$ of the manipulator’s tip along the specified path. Differentiating (7) twice with respect to time, we can derive $\ddot{\theta}$ and $\dot{\theta}$ in terms of the speed $\dot{S}$ and the acceleration $\ddot{S}$ along the path [18]. Substituting in (6) yields

$$m \ddot{S} + b \dot{S}^2 + G = T$$

(8)

where

$$m = M \theta_0$$

$$b = M \theta_0 + \dot{\theta}_0^T C \theta_0.$$  

The vectors $\theta_0$ and $\dot{\theta}_0$ represent the slope and curvature vectors of the path in joint space, respectively.

The values of $\dot{S}$ and $\ddot{S}$ are constrained by the manipulator actuator capabilities $T$. Generally, there exists at every point some speed $\dot{S}_m$ above which at least one of the actuators is saturated and the manipulator cannot stay on its specified path. Plotting $\dot{S}_m$ in the phase plane, $S - \dot{S}$ forms the velocity limit curve, where the area above the curve represents a forbidden region for the trajectory.

The time-optimal trajectory is obtained by maximizing the speed $\dot{S}$ at every point, using the maximum acceleration or deceleration at all times, so that the trajectory is below and at most tangent to the velocity limit curve [20]. Fig. 7 shows a typical time-optimal trajectory and the limit curve for the six-DOF manipulator and path shown in Fig. 6. The time obtained along such a trajectory is usually far shorter than the best feasible trapezoidal velocity profile.

The limit curve is a reflection of the actuator constraints coupled with system dynamics and path geometry. We will use it for establishing lower bound estimates for the optimal motion time along a specified path.

**E. Lower Bound Tests**

We consider four tests for lower bound estimates on the motion time along a given path, each represented by a different velocity profile. To guarantee that the estimate is a lower bound, these velocity profiles are chosen to be above the true time-optimal profile for each path, as shown schematically in Fig. 8. The estimate is computed by integrating the velocity profile assumed by each test. The tests are structured to produce successively higher lower bounds so that the velocity profile of every successive test is tangent and below the previous velocity profile. Hence

$$t_1 \leq t_2 \leq t_3 \leq t_4 = t_{op}$$

where $t_4$ is the most conservative but computationally efficient estimate, and $t_{op}$ is the optimal traveling time along the path.

1) Maximum Speed Test: The first lower bound estimate is used to select a subset of “good” paths from all possible paths along the grid. A fixed cost is assigned to every grid segment by dividing the distance along each segment by some high speed. The traveling time along a typical path is obtained by the summation

$$t_1 = \sum_j \frac{\Delta x_j}{V_{max}}$$

(9)

where $\Delta x_j$ is the length of the $j$th grid segment along the path, and $V_{max}$ is the assigned maximum speed selected as the highest speed along a velocity limit curve for a representative path. Alternatively, $V_{max}$ can be selected as the highest speed for several representative paths or the highest speed determined by the manipulator’s physical limitations.

Having assigned fixed costs to all grid segments, the best path along the grid can be found with Dijkstra’s method, and the next $k$ best paths are obtained with the Dryfus algorithm [8]. The Dryfus algorithm is similar to a shortest path search except that it effectively excludes the $k - 1$ best paths from the searched space while searching for the $k$th best path. The formal search procedure is given in Appendix C.

This test is computationally very efficient because of its simplicity. The search for a single path is proportional to the
number of nodes. The search for the \( K \) best paths is proportional to \( K \).

2) Velocity Limit Test: Once a path has been selected from the grid, it can be smoothed to eliminate the sharp corners produced by the grid segments. For this and the following tests, the grid paths are smoothed by cubic B splines, using the grid points as control points. If the smoothed path penetrates an obstacle shadow, and the path is selected as a candidate for the local optimization, the penalty function used in the local optimization will divert the path away from the obstacle as discussed later in Section III.

For this test, we assume that the speed along the path follows the velocity limit curve (see Fig. 8). The lower bound \( t_2 \) is obtained by the integral

\[
t_2 = \int_{S_0}^{S_f} \frac{1}{S_m} ds \quad (10)
\]

where \( S_m(S) \) is the velocity limit curve. The value \( t_2 \) is a true lower bound and greater than \( t_1 \) since the velocity limit curve represents the upper limit for the velocity profile along the path.

This test is more computationally demanding than the previous one but is less expensive than computing the time-optimal velocity profile. The limit curve provides a good estimate of the motion time since it takes into account the combined effects of manipulator dynamics, actuator constraints, and path geometry.

3) Maximum Acceleration, Velocity Limit, Maximum Deceleration Test: This test considers the actual speeds at the end points, which without loss of generality are chosen to be zero. Here, the manipulator accelerates from the initial point with the maximum acceleration until it reaches the limit curve; then it follows the limit curve until it switches to the maximum deceleration toward the final point. The lower bound is obtained by the summation

\[
t_3 = \int_{S_0}^{S_1} \frac{ds}{S_1(S)} + \int_{S_1}^{S_2} \frac{ds}{S_m(S)} + \int_{S_2}^{S_f} \frac{ds}{S_2(S)} \quad (11)
\]

where \( S_a \) and \( S_d \) are the speeds along the path during maximum acceleration and maximum deceleration, respectively. \( S_1 \) and \( S_2 \) are the points along the path where \( S_a \) and \( S_d \) reach the velocity limit \( S_m \). This velocity profile provides a true lower bound since it is always above the optimal trajectory. It is also below the previous profile, as shown in Fig. 8; hence, \( t_3 \geq t_2 \). The computation of \( t_3 \) is slightly higher than that of \( t_2 \) due to the need to compute and integrate the acceleration and deceleration from the end points.

4) Optimal Velocity Along the Path: This test computes the time-optimal velocity profile along the path. It is the exact solution for the optimal motion time and the upper bound for all the previous lower bound tests. The optimal velocity profile is always below the limit curve and in most cases tangent to the limit curve at a finite number of points [20].

5) Discussion: The relative effectiveness of each test depends on the computation time required and the closeness of the solution to the optimal time. The first test is the least effective in predicting motion time; however, it is the one that selects a small set of paths from the grid. The global optimality of the solution stems from the global nature of this search. The second test is better correlated with the optimal time; however, it obtains very low motion times compared to the optimal time. The third test provides a very accurate estimate of the optimal motion time by considering the acceleration and deceleration near the end points. High accelerations and decelerations near the end points are generally indicative of time-optimal paths [17].

The method presented is general and may be applicable to other dynamic systems and other cost functions. The tests outlined above are found most useful for articulated manipulators in optimizing motion time. For simple systems, such as robotic vehicles or Cartesian manipulators, the first test is found to be most effective [16].

### F. Upper Bound

The upper bound \( t_u \) is used to discard costly subsets early in the branch and bound search. To account for the improvement in the local optimization and to guarantee the global optimality of the solution, the upper bound \( t_u \) is selected below the optimal motion time along the best grid path at hand \( \bar{t} \):

\[
t_u \leq \bar{t} + \epsilon \quad (12)
\]

where \( \epsilon \) is a constant determined by the shape of the cost function near the global minimum and the grid size. To demonstrate the need for \( \epsilon \), a single dimensional cost function, evaluated at discrete points, is shown in Fig. 9. Selecting points 2, 3, 4, and 5 with costs within some threshold \( \epsilon \)
above the lowest cost guarantees that one of the selected candidates will converge in the local optimization to the global optimum. The constant \( \epsilon \) can be determined by evaluating the sensitivity of the cost function to path parameters as shown below.

Fig. 10 shows an optimal path of a two-link manipulator, represented by four control points. Fig. 11 shows the deviations from the optimal motion time along the path as functions of the perturbations of the two central control points. The curves marked \( a_i \) to \( a_4 \) correspond to variations of each control point in the \( x \) and \( y \) directions. For the case shown, motion time is most sensitive to perturbations of the left control point in the positive \( y \) direction due to the increased path curvature caused by these variations. We therefore select \( \epsilon \) based on the \( a_i \) curve. For this system, a grid size of 0.20 m would require \( \epsilon \geq 0.065 \) s.

The sensitivity test can be performed around a precomputed optimal path to be used for future optimizations of the same system between different points, or alternatively, it can be performed around the best available grid path to provide some measure of the required \( \epsilon \).

III. LOCAL OPTIMIZATION

The local optimization is used to relax the demands on the grid resolution. It obtains the time-optimal path in the vicinity of an initial guess considering manipulator dynamics, actuator constraints, obstacles, and joint limits [18]. The optimization problem is formulated as an unconstrained parameter optimization, using the control points of cubic B splines as the optimization variables and the motion time along a specified path as the cost function. Obstacles are represented by penalty functions based on the distance between the manipulator links and the obstacles. To reduce the computation time and improve the convergence of the local optimization, the number of control points is reduced by retaining only two points for each straight line segment along the grid path, as shown in Fig. 12. Grid points are eliminated only if their elimination does not force the path into obstacles. It is important to note that a small number of control points may not adequately represent the true optimal path; however, a large number of parameters is likely to divert the optimization to a numerical trap. The true optimum can be approached by successively increasing the number of control points and repeating the local optimization.

Assuming that all paths in a close neighborhood converge to the same optimum, only the best path in each region can be selected as a candidate for the local optimization. A region is defined as a tube of some radius \( D_{\text{max}} \) around a given path.

Starting with the best path obtained by the branch and bound search, all paths in a tube around the best one are discarded. These paths satisfy the inequality

\[
D = \max \left\{ \left( p_i(u) - p_i(u) \right) \right\} < D_{\text{max}},
\]

where \( p_i \) is the position vector along the best path, \( p_i \) is the position vector of any path in the remaining set of \( N - 1 \) paths, and \( u \) is a normalized path distance. This process is repeated for the next best path among the remaining paths until only paths representing distinct regions are retained. The selection of \( D_{\text{max}} \) is based on the anticipated size of the convergence region around the global optimum, obtained by the sensitivity test described earlier.

IV. IMPLEMENTATION

The method was implemented in Fortran on a \( \mu \)VAX II, for planning the motions of two- and six-DOF manipulators. The
computer program is described schematically in Fig. 13. The position space is first represented by a uniform grid (in Cartesian or joint spaces) from which all unreachable grid points due to obstacles or joint limits were eliminated. Each feasible grid segment is assigned a cost based on the first test (Section II-E.1), and the k best paths are generated. k is determined by the allowable computation time and the available computer memory. The grid paths are smoothed by B splines; then a smaller set of the best paths is retained by evaluating each path with several lower bound tests. The selected paths are filtered to select the best path in the neighborhood of each potential local optimum. These paths are then submitted to the local optimization, and the best one is selected as the global minimum.

V. EXAMPLES

The following examples demonstrate the approach for two- and six-link manipulators. The first example demonstrates the method for the two-link manipulator shown in Fig. 14 with the parameters given in Table I.

A. Two-Link Manipulator with No Obstacles

The task for the two-link manipulator is to move between the end points in minimum time. The position space, a rectangle of $4 \times 4$ m, is tessellated by a $15 \times 15$ grid, resulting in a grid size of $(400/15) = 26$ cm. Points out of reach for the manipulator tip are identified and marked with an "x", as shown in Fig. 14. The cost for the grid segments between reachable grid points is computed by dividing the segment length by the maximum speed of $V_{\text{max}} = 7$ m/s (see (9)). This value was chosen to be higher than the average highest speed along several time-optimal trajectories obtained for this manipulator.

The branch and bound search explored 714 grid paths, all shown in Fig. 15. The threshold for the upper bound $\epsilon$ was selected heuristically to be 20% above the lowest upper bound. This resulted in 96 "good" paths which after filtering, with $D_{\text{max}} = 0.30$ m, reduced to the nine paths, all shown in Fig. 16, with the times ranging from 1.166 to 1.281 s. The paths were smoothed by B splines with 4 to 6 control points (8 to 12 parameters). These paths converged in the local optimization to the optimal path A shown in Fig. 17, with the traveling time of 1.10 s. This path is consistent with results obtained by other intuitive methods [17].

The improvement in motion time from the best grid path to the optimal one was $1.166 - 1.10 = 0.066$ s. This is lower than the upper bound on the improvement of 0.09 s suggested by the sensitivity curves in Fig. 11 for a grid size of 0.26 m. This confirms that the $\epsilon$ used in this optimization was sufficiently high to guarantee convergence to the global optimum. Fig. 17 shows another local optimal path, marked $B$, with the time of 1.34 s. No path in this region was selected by the branch and bound search because of their high cost relative to paths in the upper region. Note that the optimization was carried out in the Cartesian space, considering only one kinematic solution and avoiding kinematically singular points. This excluded the global optimal path that was shown to pass through the manipulator base [10].

![Fig. 13. Global search for the optimal path.](image)

![Fig. 14. A two-link manipulator at the end points.](image)

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PARAMETERS OF THE TWO-LINK MANIPULATOR</strong></td>
</tr>
<tr>
<td>$L_1 = 1.0$ m, $L_2 = 0.5$ m, $m_1 = 1$ kg, $f_1 = 0.2$ kg-m, $T_1 = 10$ N-m</td>
</tr>
<tr>
<td>$L_2 = 1.0$ m, $L_2 = 0.5$ m, $m_2 = 1$ kg, $f_2 = 0.3$ kg-m, $T_2 = 10$ N-m</td>
</tr>
</tbody>
</table>

The relative effectiveness of the various tests is shown in Fig. 18. Each curve shows the costs for all the paths evaluated by a particular test in a ranked order. The grid search selected the best 714 paths, out of which 686 were selected by the second test. The number of paths was reduced to 451 and 96 by the third and fourth tests, respectively. The 96 paths were reduced to 9 paths, all submitted to the local optimization. The computation time for the branch and bound search was about 3 min of CPU time, and for the local optimizations about 20 min. Using a larger $D_{\text{max}}$, would retain only one path, reducing computation time to about 5 min. The low computation time of the local optimization is attributed to the closeness of the initial guess to the optimal solution. The time spent to obtain a good initial guess with the branch and bound search was, therefore, well worth it.
Fig. 19 shows a case where the optimization converged to two optimal paths. In this case, the end points are antisymmetric relative to the center, and as expected, there are two antisymmetrical optimal solutions with similar traveling times. The times are not identical for both paths because of the inaccuracies associated with the numerical optimization.

**B. Two-Link Manipulator with Obstacles**

The task for the manipulator is to move between the same end points as in Fig. 14 in minimum time while avoiding the obstacles shown in Fig. 20. Fig. 20 shows also the obstacle shadows, assuming that the links and the obstacles are in the same horizontal plane. The grid points inside the obstacle shadows are marked by small crosses; points out of reach or points of singularity are marked by `×`'s. A higher resolution grid was used to verify that the $15 \times 15$ grid sufficiently represents the obstacles. The consideration of only one kinematic configuration is consistent with the desire to avoid singularities along the path. For the case shown, this does not represent a problem since an obstacle-free path exists without the need to change configurations.

The initial graph search produced for this case 238 feasible paths. This is less than that for the case with no obstacles since many paths were eliminated because of their collision with obstacles. The branch and bound search produced 57 paths with times ranging from 1.471 to 2.07 s, all shown in Fig. 21. Using $D_{\text{max}} = 0.25$ m, only one path remained, with a traveling time of 1.471 s. Submitting this path to the local optimization resulted in the optimal path, shown in Fig. 22, with an optimal time of 1.416 s. The computation time for the grid search was about 3 min of CPU time, and for the local optimizations about 10 min.

**C. Three-Dimensional Paths for a Six-DOF Manipulator**

The following examples demonstrate the feasibility of this method for a six-DOF manipulator moving in a three-dimensional space. The manipulator parameters are given in Table II. In this example, the motions of the first three joints are being optimized, while the wrist joints are fixed at a specified orientation. The dynamics of the full six-DOF manipulator and the loads on the wrist actuators are taken into account.

The task for the manipulator is to move from work station A to work station B, shown in Fig. 23. The work space was tessellated by a $10 \times 10 \times 10$ grid, with a grid size of 0.4 m. Two paths, with traveling times of 0.586 and 0.604 s, were selected out of the best 560 grid paths, using $\epsilon = 10\%$ of the best time, and $D_{\text{max}} = 0.4$ m. Both paths converged in the local optimization to almost the same path with traveling times of 0.430 and 0.431 s. The convergence to the same path indicates that the region of convergence around the global optimal path is relative large for this system. The computation time was in the order of 2 h on the µVAX II. Most of the computation time was spent on the local optimization. The search and filtering process required about 20 min.

The second example demonstrates time-optimal motions of the six-DOF manipulator between the same end points as in the previous example while avoiding the obstacles C and D.
The geometry of the path obtained by the local optimization reflects a specific choice of the penalty function for each obstacle. By selecting the weighting factors for the penalty function of each obstacle, it is possible to divert the path away from specific obstacles and allow it to come closer to others. In this example, the penalty for obstacle $D$ was selected five times higher than that for the other obstacles.

VI. Conclusions

A method has been presented for computing the optimal motion of a manipulator, considering its dynamics, actuator constraints, joint limits, and obstacles. It combines a search for a set of the best smooth paths from a tessellated position space with a local path optimization, eliminating the need for an initial guess. The approach consists of evaluating a large number of paths in the position space for their optimality (motion time) using a branch and bound search and a series of lower bound estimates on the traveling time along each path. The reduction of the search to the position space and the choice of the estimates, based on manipulator dynamics and the characteristics of their time optimal motions, makes this approach computationally efficient relative to other methods that obtain global optimal motions, such as a dynamic programming search in the state space. The complexity of this approach arises exponentially with the number of DOF's and grid resolution and is linear in the number of best paths selected from the initial set. The computation of the local optimization is polynomial in the number of degrees of freedom and the number of control points selected to represent the path [24].

Obstacles and joint limits are considered by eliminating the obstacles and their shadows from the searched space. Interestingly, the consideration of obstacles makes the approach more efficient as it reduces the number of paths to be considered. For this reason, confined spaces are easier to handle in the first global search than sparsely populated environments. The increased number of control points, however, required to represent an admissible path in tight spaces may increase the computation of the local optimization.

The grid size of the tessellated space and the number of paths in the first set are important factors in determining the global optimality of a solution. The solution obtained by the branch and bound search is a global optimum if the grid is sufficiently small and the upper bound is appropriately chosen. The requirement on grid resolution is relaxed by assuming that the region of convergence around the optimal path is large compared to the grid size. A simple sensitivity test is proposed to determine the appropriate threshold above the upper bound for a given grid resolution.

Examples are presented that demonstrate the approach for two- and six-DOF manipulators. The solutions obtained for the two-link manipulator with and without obstacles are only near-global minimum since the optimization in the Cartesian space avoids singularity points and the number of control points representing the path might be too small. These results, however, are consistent with an intuitive method that allows predicting the general shapes of time-optimal paths [17].

shown in Fig. 24. Here, the initial search resulted in 380 paths, which reduced before filtering to 65, using the same parameters as in the previous example. After filtering, only two paths remained with the times of 0.738 and 0.750 s. The path with the time of 0.738 s converged to the optimal path shown in Fig. 24, with an optimal time of 0.51 s. The large difference between the times for the paths before and after the local optimizations indicate that the grid size was too large for these cases. The grid size was limited by the computer memory allocated on the $p$VAX II. The computation times for this case were of the same order as for the previous example.
TABLE II
PARAMETERS OF THE SIX-DOF MANIPULATOR

<table>
<thead>
<tr>
<th>Link: 1</th>
<th>( L = 1.00 \text{ m}, ; L_{cg} = 0.25 \text{ m}, ; \text{Mass} = 7.0 \text{ kg} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Principal Moments of Inertia (N-m/s²)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Link: 2</td>
<td>( L = 1.00 \text{ m}, ; L_{cg} = 0.30 \text{ m}, ; \text{Mass} = 4.0 \text{ kg} )</td>
</tr>
<tr>
<td></td>
<td>Principal Moments of Inertia (N-m/s²)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Link: 3</td>
<td>( L = 0.25 \text{ m}, ; L_{cg} = 0.25 \text{ m}, ; \text{Mass} = 3.0 \text{ kg} )</td>
</tr>
<tr>
<td></td>
<td>Principal Moments of Inertia (N-m/s²)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Actuator Torques:

<table>
<thead>
<tr>
<th>Min.</th>
<th>Max. (N-m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-300.00</td>
<td>300.00</td>
</tr>
<tr>
<td>-200.00</td>
<td>200.00</td>
</tr>
<tr>
<td>-100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>-50.00</td>
<td>50.00</td>
</tr>
<tr>
<td>-50.00</td>
<td>50.00</td>
</tr>
<tr>
<td>-50.00</td>
<td>50.00</td>
</tr>
</tbody>
</table>

Fig. 22. The manipulator at various points along the optimal path.

Fig. 23. The optimal path of a six-DOF manipulator without obstacles.

Fig. 24. The optimal path of a six-DOF manipulator with obstacles.

The solutions obtained for the six-DOF manipulator were obtained by specifying the orientation of the end-effector along the path, reducing the optimization problem to a three-dimensional optimization. Although only the first three joints were optimized, the dynamics of the entire six-DOF system were considered. The optimal solution obtained is therefore consistent with the dynamics of the full system, guaranteeing that no actuator exceeds its limits. The results cannot be compared to any existing work as we are not aware of any method that has been successfully demonstrated to problems of similar magnitude.

APPENDIX A

BRANCH AND BOUND SEARCH

Let

\[ P_0 = \text{the initial set of all feasible paths}; \]
\[ P_k = \text{a subset of } P_{k-1}; \; k = 1, 2, \ldots; \]
\[ t_u = \text{upper bound on the optimal time}; \]
\[ t(p) = \text{the optimal traveling time along a given path, } p \in P_0; \]
\[ t_i(p) = \text{lower bound on the traveling time along path } p, \]

produced by the \( i \)th test

\[ t_i(p) < t_{i+1}(p), \; \; i = 1, q \]
\[ t_q(p) = t(p) \]

Step 0: (start)
Set the upper bound \( t_u = \infty \)
Set \( k = 1 \)

Step 1:
select \( k \)th best path \( p_k \) in \( P_{k-1} \) by computing \( t_k(p_k) \),
open node \( p_{1,k} \) with cost \( t_1(p_k) \)
subtract \( p_k \) from \( P_{k-1} \) and open node \( p_k \) with cost \( t_k(p_k) \)
close node \( P_{k-1} \)

Step 2:
if all nodes are closed, go to Step 3
find best open node
if best node is \( P_k \):
set \( k = k + 1 \)
go to Step 1
else: best node is \( p_{i,j} \)
compute \( t_{i,k}(p) \),
close node \( p_{i,j} \) and open node \( p_{i+1,j} \) with cost \( t_{i+1}(p) \)
if \( t_{i+1}(p) > t_u \) close node \( p_{i+1,j} \)
if \( i + 1 = q \) close node \( p_{q,j} \) and mark it as a solution
if \( t_q(p) < t_u \) set \( t_u = t_q(p) \)
go to Step 2

Step 3:
Select the optimal path
Stop.///

APPENDIX B
GLOBAL OPTIMALITY OF THE BRANCH AND BOUND SEARCH

We prove that the branch and bound search does converge to the global optimal path if one was included in the initial set \( P_0 \).

Let \( P \) be the set of all possible paths between given end points, and \( P_0 \subset P \) be the set of all feasible paths defined by the grid. The function \( t_i(\cdot) \), \( i = 1, \ldots, 4 \), returns a lower bound estimate of order \( i \) on the motion time along a given path \( p \in P_0 \), and

\[
t_1 \leq t_2 \leq t_3 \leq t_4 = t_{opt}
\]

(\( B1 \))

where \( t_{opt} \) is the true optimal motion time along the path.

Define the global optimal path \( \tilde{P} \) and the optimal time \( \tilde{t} \):

\[
\tilde{t} = t_{opt}(\tilde{P}) = \min \{ t_{opt}(p), p \in P_0 \}
\]

(\( B2 \))

and

\[
P_{k+1} = P_k - p_{k+1}
\]

(\( B3 \))

where

\[
t_i(p_{k+1}) = \min \{ t_i(p), p \in P_k \}.
\]

The set of the selected paths \( \tilde{P}_k \) is

\[
\tilde{P}_k = P_0 - P_k = \{ p_j, j = 1, \ldots, k \}
\]

(\( B4 \))

Note that \( p_k \in P \) and \( p_k \notin \tilde{P}_k \).

From (\( B1 \)), it follows that \( t_i(p_k) \) is the lower bound on the set \( P_k \):

\[
t_i(p_k) \leq t_i(p), p \in \tilde{P}_k \}
\]

(\( B5 \))

The search terminates when the upper bound \( t_u \) on the optimal motion time is greater or equal to the lower bound on \( \tilde{P}_k \) for some \( k \):

\[
t_u = \min \{ t_{opt}(p), p \in \tilde{P}_k \} \leq t_i(p_k), p \in \tilde{P}_k.
\]

(\( B6 \))

From (\( B5 \)) and (\( B6 \)), it follows that

\[
\min \{ t_{opt}(p), p \in P_k \} \geq t_i(p_k) \geq \min \{ t_{opt}(p), p \in \tilde{P}_k \}.
\]

(\( B7 \))

and (\( B4 \)) implies that

\[
\min \{ t_{opt}(p), p \in \tilde{P}_k \} = \min \{ t_{opt}(p), p \in P_0 \} = \tilde{t}.
\]

(\( B8 \))

This completes the proof.///

This proof shows that, if the optimal path was included in the initial set \( P_0 \), it will be found by the branch and bound search when the search is terminated. We modify the proof to account for the case where the initial set includes only a near-optimal path.

The optimal time obtained by the branch and bound search \( \tilde{t} \) is only a near-minimum time. It can be further optimized by the local optimization \( LO(p) \). If the global optimal time is \( t^* \), then

\[
t^* = \min \{ t | t = LO(p), p \in P_0 \}
\]

(\( B9 \))

assuming that \( P_0 \) spans all the different possible local optimal regions.

Define \( \epsilon \) as a upper bound on the time difference, obtained by the local optimization, between the global optimum \( t^* \) and the nearest grid path \( t \). The difference between \( t^* \) and the best grid path \( \tilde{t} \), defined in (\( B2 \)), is therefore also bounded

\[
\tilde{t} - t^* = \delta \leq \epsilon.
\]

(\( B10 \))

Define a set of near-optimal paths \( \tilde{P} \)

\[
\tilde{P} = \{ p | t_i(p) < \tilde{t} + \epsilon, p \in P_0 \}.
\]

(\( B11 \))

We need to show that, for a given \( \epsilon \) and \( \tilde{t} \), the path with the global optimal time \( t^* \) is obtained from one of the paths included in the set \( \tilde{P} \)

\[
t^* = \min \{ t | t = LO(p), p \in \tilde{P} \}
\]

(\( B12 \))

Assume that a path \( p^* \in \tilde{P} \) converged to the optimal time \( t^* \), i.e., \( t^* = LO(p^*) \). Its time before the optimization is \( t_i(p^*) \). Then, from (\( B10 \)),

\[
t_i(p^*) \leq t^* + \epsilon \leq \tilde{t} + \epsilon.
\]

(\( B13 \))

But, if the path \( p^* \), satisfies (\( B13 \)), then from (\( B11 \)) \( p^* \in \tilde{P} \), which is a contradiction. This proves (\( B12 \)).///

APPENDIX C
DREYFUS METHOD FOR K SHORTEST PATHS

The algorithm for the \( K \) best paths is due to Dreyfus [8]. The algorithm finds the \( K \) shortest paths from the origin to each of the \( n - 1 \) nodes of the network, and is of \( O(Kn^2) \) complexity. The first optimal path is obtained with a Dijkstra best path search method [8]. The next best paths are obtained as follows:

Let \( u_j^{(m)} \) be the cost of the \( m \) shortest path from the origin to node \( j \) and let \( \mu(k, j, m) \) be the number of times that edge \( (k, j) \) was the final edge in the set of the \( m \) shortest paths to node \( j \). The cost to node \( j \) of the \( m + 1 \) shortest path is updated by

\[
u_j^{(m+1)} = \min \{ u_k^{(m)}(k,j) + a_{kj} \}, \quad k = 1, \ldots, q
\]

(\( C1 \))

where \( a_{kj} \) is the cost over the edge \( (k, j) \), and \( q \) is the number of nodes having an edge to node \( j \). \( \mu \) is incremented for \( k = i \) that minimizes (\( C1 \)):

\[
\mu(i, j, m + 1) + \mu(i, j, m) + 1
\]

\[
\mu(k, j, m + 1) + \mu(k, j, m), \quad k \neq i
\]

(\( C1 \))

The cost at all nodes for the first path \( u_j^{(1)} \) is initialized with Dijkstra's method, and the counter \( \mu \) is set to

\[
\mu(i, j, 1) = 1, \quad \text{for all } j, \text{ and the optimal direction } i
\]

\[
\mu(k, j, 1) = 0, \quad \text{for all } j, \text{ and all } k \neq i.
\]
The $m + 1$ path is obtained after the $m$ paths to all nodes are known. Note that if $\mu(k, j, m) = m$ on the right side of (C1), the cost $u^{[m+1]}$ may depend on the $m + 1$ cost to another node $u^{[m]}$. Therefore, the order in which the nodes are processed is important in order for $u^{[m+1]}$ to be known when needed. Lawler [8] suggests to process the nodes in the order of the number of edges in their current paths. Processing the nodes in the order of lowest cost is, however, more efficient [15].

REFERENCES


Zvi Shiller (S'85 - M'87) received the B.Sc. degree from Tel Aviv University, Tel Aviv, Israel, in 1976, and the M.S. and Sc.D. degrees from M.I.T., Cambridge, MA, in 1984 and 1987, all in mechanical engineering.

Since 1987, he has been an Assistant Professor in the Mechanical, Aerospace and Nuclear Engineering Department at the University of California, Los Angeles. Between 1976 and 1982, he worked as a design engineer for the Israeli Aircraft Industries, Israel. His research interests include motion planning and control of computer-coordinated mechanical systems, mechanical design, multidisciplinary dynamics, and autonomous vehicles.

Steven Dubowsky (M'72) received the Bachelor’s degree from Rensselaer Polytechnic Institute, Troy, NY, in 1963, and the M.S. and Sc.D. degrees from Columbia University, New York, NY, in 1964 and 1971, respectively.

He is currently a Professor of Mechanical Engineering at the Massachusetts Institute of Technology, Cambridge. He has been a Professor of Engineering and Applied Science at the University of California, Los Angeles; a Visiting Professor at Cambridge University, Cambridge, England; and a Visiting Professor at the California Institute of Technology, Pasadena. During the period from 1963 to 1971, he was employed by the Perkin-Elmer Corporation, the General Dynamics Corporation, and the American Electric Power Service Corporation. His research has included the development of modeling techniques for manipulator flexibility and the development of optimal and self-learning adaptive control procedures for rigid and flexible robotic manipulators. He has authored or coauthored nearly 100 papers in the area of the dynamics, control, and design of high-performance mechanical and electromechanical systems.

Prof. Dubowsky is a Registered Professional Engineer in the State of California and has served as an advisor to the National Science Foundation, the National Academy of Science/Engineering, the Department of Energy, and the U.S. Army. He is a Fellow of the ASME and is a member of Sigma Xi and Tau Beta Pi.