Robotica (1997) volume 15, pp 555–562. Printed in the United Kingdom © 1997 Cambridge University Press

Minimum cost, fixed time trajectory planning in robot manipulators. A suboptimal solution

Ignacy Dulęba

Institute of Engineering Cybernetics, Technical University of Wrocław, Janiszewskiego 11/17, 50–372 Wrocław (Poland)

(Received in final form: December 3, 1996)

SUMMARY

In this paper the minimum cost trajectory planning problem with fixed time in robot manipulators is considered. The task is solved by transforming the problem to a set of free right-end time optimal problems, leading to a suboptimal solution. Each problem of the optimal cost trajectory planning with a free time is effectively solved by the method of minimal neighbourhood. An algorithm for the task of suboptimal cost trajectory planning with fixed time is presented and applied to the model of a PUMA-like robot. Results of the paper seem to be of particular relevance to the optimization of multi-robot systems.

KEYWORDS: Robot manipulator; Trajectory planning; Optimization; Multi-robot systems.

1. INTRODUCTION

The aim of modern manufacturing is to produce high quality goods fast and cheap. The high quality of robot-made goods is usually guaranteed by repeatability of robot motions, while fast and cheap production requirements are satisfied by applying some optimization techniques using a performance index defined appropriately. The usefulness of robots is increased when robots collaborate with each other in a production line. Unfortunately, there is a gap in robotic literature concerned with problems appearing at the border line between single-robot and multi-robot systems. Optimal tasks for one robot have been stated and solved: For a prescribed path the robot has to follow, and the minimum time criterion,[1–4] for a prescribed path, and the minimum cost criterion,[5–8] for prescribed both the path and the time while minimizing cost.[9] On the other hand, techniques worked out by operation research theory are applicable to multi-robot systems only when elementary tasks are equipped with some number characteristics (e.g. time of operation completing, energy spending etc.). The gap, mentioned above, relies upon inconsistency of output data generated by one-robot problems with input data for optimizing techniques of multi-robot systems. Tasks for one robot are usually formulated without imposing any condition on time (with free right-end time) of their completion, while data for multi-robot optimizing methods need to be more time-flexible, i.e. they should allow for conditions like end-time prescribed, interval prescribed, etc.). As an example of the inconsistency let us consider a two-robot system when tasks for both of the robots are planned independently although optimally, according to a criterion common to each robot and time free. If prescribed times of performed tasks differ significantly and a rest value of the criterion for the robot (i.e. value of the criterion spent on taking the robot at rest) which first finishes its motion is big, the resulting solution is usually far from optimal for the pair of robots. The aim of this paper is to fill partially the gap. We will achieve it by considering a suboptimal cost trajectory planning task with the time fixed. Some comments will be made on the case while conditions imposed on time are not as strict as in the case under consideration, raising the task with free right-end time.

To our knowledge the cost-optimal fixed time trajectory planning task along prescribed paths has been dealt with in robotic literature only in reference 9 by applying Bellman's dynamical programming (DP) in an explicit time-dependent space.

However, the approach presented above has few serious drawbacks. They are as follows: 1. admissibility of controls is checked only in discrete-time locations; 2. the velocity $\dot{q}$ is approximated in a rather rough manner; 3. a regular division of the time axis may lead to an unregular distribution of points along a prescribed path. Hence, in the latter case coefficients of the dynamics may differ significantly over step-time interval leading to the model inaccuracy. In this paper we propose an approach free from the drawbacks.

Our task can be formulated in terms of the Pontriagin's maximum principle as well. However, the inefficiency of the classical Pontriagin's approach is caused by the highly coupled and nonlinear robot dynamics. Still, a couple of years ago the only effective method for solving the minimum cost trajectory planning with free right-end time was DP applied in a space that explicitly did not depend on time.[5] There to apply DP the problem has been transformed into the phase-plane (s, μ) where s is a parameter describing a path to follow and μ its time derivative, after discretizing the phase-plane the solution is obtained in the well known way. Recently, new methods based on the discretized phase-plane technique have been introduced[7,8] in order to obtain the solution more effectively than in the DP case. They are, in a chronological order, a method based on the graph-searching technique,[7] and the most effective up to now – the method of minimal neighbourhood.[8] It should be noticed that the minimum cost trajectory planning tasks with time fixed are much more involved

https://doi.org/10.1017/S0263574797000647 Published online by Cambridge University Press

than with right-end free time. The reason lies in difficulties with introducing a fixed time condition into DP method applied to the phase-plane. The same problem is faced when two other new methods, mentioned above, are employed in the phase-plane framework. The standard approach in optimization techniques and in the calculus of variations[10] when an inconvenient integral condition appears is to plug the function under this integral premultiplied by a Lagrange multiplier into the criterion function for the task considered. Then, we solve a set of elementary tasks with the Lagrange multiplier varied, without the inconvenient condition and the modified criterion function, until the multiplier reaches such a value which allows to satisfy the inconvenient condition. Obviously, the approach is as effective as a method solving the elementary task (when the Lagrange multiplier is fixed). The main achievement of this paper is to show that the method of minimal neighbourhood[8] is a really good tool in solving the fixed-time task. Especially, that its efficiency is tightly related to an initial guess of a trajectory optimized in an iterative process. In solving the fixed-time task when two successive free-time tasks are to be solved with the Lagrange multiplier modified slightly the resulting trajectory of the first task can serve as a very good initial trajectory for the second task.

Moreover, the method of minimal neighbourhood is also useful in a cost optimal task with the time of its completion not so strictly defined as in the fixed-time case and in the task of optimizing a motion of a two-robot system.

This paper is organized as follows: In Section 2 the trajectory planning task with fixed time is formulated in terms of the phase-plane and the problem transformation is discussed. Section 3 is devoted to an algorithm solving the task and to a classification of solution types. Section 4 contains two tasks carried out on the computer model of a PUMA-like robot, exemplifying the main solution types. Section 5 presents applications of the developed algorithm. The paper is concluded with Section 6.

2. MINIMUM COST TRAJECTORY PLANNING WITH FIXED TIME IN THE DISCRETIZED PHASE-PLANE: TASK FORMULATIONS

The trajectory planning problem amounts to finding the time history of robot joint positions, velocities and accelerations. When the path to be followed is prescribed and the transformation from external, Cartesian space, where usually motion tasks are formulated, to internal, joint space, is unique, only the time parameterization of the path is looked for. We assume that the path is described by a set of n functions smooth enough to comply with conditions imposed further on:

$$q(s) = (q_i(s) \mid i = 1 \cdots n)^T, \quad s \in [0, s_{\text{fin}}], \tag{1}$$

where s denotes a scalar (the arc length along the path in our case) parameterizing the desired path and n is the number of DOF. The initial and the final configurations are achieved, respectively, at $s = 0$ and $s = s_{\text{fin}}$. Time derivatives of the joint variables can be expressed as follows:

$$\begin{aligned} \frac{dq_i}{dt} = \dot{q}_i &= \frac{dq_i(s)}{ds} \cdot \frac{ds}{dt} = \frac{dq_i(s)}{ds} \cdot \mu, \\ \frac{d^2q_i}{dt^2} = \ddot{q}_i &= \frac{dq_i(s)}{ds} \cdot \dot{\mu} + \frac{d^2q_i(s)}{ds^2} \cdot \mu^2, \end{aligned} \tag{2}$$

where $\mu = \dot{s}$ can be treated as the velocity measured along the path. When the derivatives $\frac{dq_i}{ds}$ and $\frac{d^2q_i}{ds^2}$ exist the robot dynamics expressed in closed form as:[11]

$$M_{ij}(q)\ddot{q}_j + C_{ijk}(q)\dot{q}_j\dot{q}_k + R_{ij}\dot{q}_j + G_i(q) = u_i \mid i = 1, \ldots, n, \tag{3}$$

can be transformed, by plugging (1) and (2) into (3), to the form:[5]

$$\begin{cases} \dot{s} = \mu \quad \mid i = 1, \ldots, n \\ M_{ij}(s) \cdot \dfrac{dq_j}{ds} \cdot \dot{\mu} + M_{ij}(s) \cdot \dfrac{d^2q_j}{ds^2} \cdot \mu^2 \\ \qquad + C_{ijk}(s) \cdot \dfrac{dq_j}{ds} \cdot \dfrac{dq_k}{ds} \cdot \mu^2 \\ \qquad + R_{ij} \cdot \dfrac{dq_j}{ds} \cdot \mu + G_i(s) = u_i \end{cases} \tag{4}$$

where: u_i is the torque/force applied at the i-th joint; M_{ij} is the inertia matrix, C_{ijk} is the Coriolis and centrifugal force array, R_{ij} is the matrix representing viscous friction, G_i is the gravitational force. For equations (3), (4) the Einstein summation convention is used, and dummy indices run from 1 to n. By grouping terms in the equations (4) the model of dynamics in the phase-plane is described by the formula:

$$u_i = \tilde{M}_i(s)\dot{\mu} + Q_i(s)\mu^2 + \tilde{R}_i(s)\mu + G_i(s) \mid i = 1, \ldots, n. \tag{5}$$

The performance index evaluating the cost of motion will be assumed in the form:

$$J(u(\cdot)) = \int_0^T L(q, \dot{q}, u)\, dt, \quad \text{with } T \text{ fixed}, \tag{6}$$

while a feasibility condition on controls is given by the formula:

$$u \in E(q, \dot{q}) = \tilde{E}(s, \mu). \tag{7}$$

Taking into account the formula for the current time:

$$t(s) = \int_0^s \frac{1}{\mu(\Theta)}\, d\Theta, \tag{8}$$

and after plugging (1), (2) and (5) into (6) the performance index expressed in phase-plane terms is as follows:

$$J(u(\cdot)) = \int_0^{s_{\text{fin}}} \tilde{L}(s, \mu, u(s, \mu, d\mu/ds))\, ds, \tag{9}$$

https://doi.org/10.1017/S0263574797000647 Published online by Cambridge University Press

with an equality condition imposed on the right-end time:

$$T = \int_0^{s_{\text{fin}}} \frac{1}{\mu}\, ds. \tag{10}$$

Assuming that the robot begins and ends its motion at rest:

$$\dot{q}(0) = \dot{q}(T) = 0 \;\Rightarrow\; \mu(0) = \mu(s_{\text{fin}}) = 0 \tag{11}$$

the task with fixed right-end time can be formulated as follows:

Task (*) Find a trajectory in the phase-plane which follows the prescribed path given by Eq. (1) and minimizes the functional (9) while conditions (4), (7), (10), (11) are met.

When a trajectory solving the Task (*) is found, all quantities depending on the parameter s can be easily expressed as functions of time according to formula (8). Although the task has only 2 independent variables (s, μ), it is usually still too difficult to be solved. So, the phase-plane is covered by a (Δs, $\Delta\mu$) grid and a model of motion between two points (called later nodes) of the grid is also assumed to ease the task. The assumption we impose here following[5] is the constancy of the pseudo-acceleration ($\dot{\mu}$) on each interval $[s_k, s_{k+1}]$. It is known that in this case the pseudo-veloxity is described by the formula:[5]

$$\mu(s) = \sqrt{\frac{(s_{k+1} - s)\cdot \mu_k^2 + (s + s_k)\cdot \mu_{k+1}^2}{(s_{k+1} - s_k)}} \tag{12}$$

where μ_k denotes $\mu(s_k)$ and $\mu_{k+1} = \mu(s_{k+1})$, and the controls are as follows:

$$u(s) = Q\mu^2(s) + \tilde{R}\mu(s) + G + \tilde{M}\cdot\frac{\mu_{k+1}^2 - \mu_k^2}{2(s_{k+1} - s_k)} \quad \text{for } s \in [s_k, s_{k+1}]. \tag{13}$$

with Q, $\tilde{R}$, $\tilde{M}$, G defined by (4), (5). This assumption has not been made by chance. It makes checking the feasibility condition on controls particularly easy because of the simple, quadratic dependence of u on μ.

After discretizing, the Task (*) should be reformulated. Let us observe that the discretization has moved the domain of sought for solutions (functions $t(s)$) from the space of strictly monotonic functions on the interval $[0, s_{\text{fin}}]$ to the finite space of all sequences $(s_i, \mu_j) \,|\, i = 0, \ldots, m$ where m is the number of intervals into which the parameter s is divided, and all μ_j are bounded. This implies that Eq. (10) may not be satisfied, so the condition (10) has to be weakened to ensure the solution existence. The solution will be acceptable if the motion times obtained and prescribed differ by less than a prescribed value. Now we are in a position to state a Task (**) in the discrete phase-plane terms.

Task (**) Find a sequence of nodes (fully determining the trajectory) in the phase-plane which follows the prescribed path given by Eq. (1) and minimizes the functional (9) while conditions (4), (7), (11) are met along with

$$\left| \int_0^{s_{\text{fin}}} \frac{1}{\mu}\, ds - T \right| < \epsilon. \tag{14}$$

with a prescribed $\epsilon > 0$. The value of ε should be selected according to the Δs, $\Delta\mu$ values. The rule is: the finer the grid, the smaller ϵ.

To solve the Task (**) the inconvenient condition (14) must be restated in such a way that DP-like methods on the phase-plane would be applicable. The idea we propose relies upon replacing the criterion function by another Lagrange multiplier dependent function, solving the Task (**) without condition (14) in terms of the multiplier and choosing such its value that realizes the robot motion in the prescribed time. The methodology just presented will be called later on **the time-matching**.

The criterion function in the phase-plane terms described by Eq. (9) is now replaced by a new parameter-dependent[10] criterion function to take into consideration the integral condition (10):

$$J_\lambda(u(\cdot)) = \int_0^{s_{\text{fin}}} \left(\lambda \cdot \frac{1}{\mu} + \tilde{L}(s, \mu, u(s, \mu, d\mu/ds)) \right) ds, \tag{15}$$

where λ is a parameter (a Lagrange multiplier) to be determined. It may vary but there exists such a $\lambda = \lambda^*$ (unless T is shorter than the minimal time used to follow the path) which satisfies the condition:

$$\left| \int_{s=0}^{s_{\text{fin}}} \frac{1}{\mu^*(s)}\, ds - T \right| = |T(\lambda^*) - T| < \varepsilon. \tag{16}$$

A Task (***) we are going to solve later on is as follows:

Task (***) Find a sequence of nodes in the phase-plane and a value of λ^*, where the prescribed path given by Eq. (1) is followed, minimizing the functional (15) while conditions (4), (7), (11), (14) are met.

As the solution we treat the sequence of nodes, the λ^* value and λ^*-independent part of the functional (15):

$$J(u(\cdot)) = \int_0^{s_{\text{fin}}} \tilde{L}(s, \mu, u)\, ds = J_{\lambda^*}(u(\cdot)) - \int_0^{s_{\text{fin}}} \frac{\lambda^*}{\mu(s)}\, ds. \tag{17}$$

Having formulated the Task (***), we are going to investigate its solution quality compared with that of the Task (**). It is obvious that the solution of the Task (***) cannot be better than that of the Task (**), because a trajectory generated by the Task (***) is among trajectories satisfying necessary condition (14), whereas the best trajectory (in the sense of criterion (9)) is just the solution of the Task (**). Nevertheless, we expect that the solution of the Task (***) is suboptimal, i.e. a trajectory generated by solving the Task (***) will produce a value of performance index (9) less than for a majority of trajectories satisfying condition (14).

https://doi.org/10.1017/S0263574797000647 Published online by Cambridge University Press

Arguments supporting such an expectation are as follows:

- the method based on the analogue of the Lagrange multiplier technique[10] guarantees at least a locally optimal solution when a regularity condition is met

$$\frac{\partial G}{\partial \mu} - \frac{d}{ds}\frac{\partial G}{\partial \mu'} \neq 0 \tag{18}$$

where $G(s, \mu, \mu' = d\mu/ds)$ is an under-integral function in an integral-type constraint. In our case $G(s, \mu, \mu') = \frac{1}{\mu}$ (cf. Eq. (10)) and the regularity condition (18) holds besides boundary points,
- if the prescribed time T is realized by only one trajectory (eg. min-time), the sub-optimal solution is optimal as well;
- if the prescribed time is close to the time realizing minimum cost trajectory planning with free time, the suboptimal and optimal solution are the same, because in this case the value of λ approaches zero.
- the approach eliminates trajectories which satisfy condition (14) and exhibit abrupt changes in the phase-plane. The elimination takes place with regard to the cost spent on jumps $\left(\text{cf. formulae } \dot{\mu} = \frac{d\mu}{ds} \cdot \mu \text{ and (5), (9)}\right)$;

Now a rough description of core steps of an algorithm solving the Task (***) can be given. The algorithm relies on generating purposefully a sequence of λ until the condition (14) is met (in the case $\lambda = \lambda^*$). More precisely, an interval for λ containing λ^* is established and each next probe value of λ shrinks the interval keeping λ^* in its interior. It is easy to observe that the iterative process needs an effective method for solving the cost optimal task with free time (the right-end time in the Task (***) is used only for checking if the solution is admissible). Fortunately, the method of minimal neighbourhood satisfies the requirement. Below, the main features of the method are presented while details have been given in reference 8. The method works in the discretized phase-plane and relies upon an admissible trajectory neighbouring, solving the task in the neighbourhood by the dynamical programming and modifying the neighbourhood. Two last steps are applied repeatedly until the stop condition is not satisfied, i.e. the resulting trajectory for the current iteration does not touch the border of the neighbourhood for the iteration. Efficiency of the method depends on some number, called the rank,[8] which controls the size of the area considered in the current iteration, and the distance (the area in the phase plane) between the initial and the resulting trajectories. In our case the second property is particularly important. During the iterative process of generating λ, when two subsequent values of λ differ slightly, the resulting trajectory for a given λ can be used as the initial one for the next task speeding-up the computation. Numerical examples presented in Section 4 are in favour of this observation. In the next section details of the algorithm for the Task (***) will be presented and a classification of solution types proposed.

3. AN ALGORITHM OF SUBOPTIMAL COST TRAJECTORY PLANNING WITH FIXED TIME

A sketch of the algorithm of solving the Task (***) has been given in the previous section. Now we will make it more specific and then describe the steps of the algorithm in detail. It is obvious that the solution type is greatly influenced by the value of the prescribed time of motion (T). Below several particular cases are distinguished with respect to T varying from 0 to infinity. These cases are as follows:

(i) $0 \leq T \leq T_t$ there is no solution; here T_t denotes the time for the minimum time trajectory planning.

(ii) $T_t \leq T \leq T_c$ the robot does not stop, solution at the phase-plane lies somewhere between the minimum time and the minimum cost with no condition imposed on time; here T_c denotes the time for the minimum cost trajectory planning with free right-end time.

(iii) $T_c \leq T \leq T_{sl}$ the robot may stop at a suitable location along the path; here T_{sl} denotes the maximum admissible time for performing the task without robot stopping anywhere. For the $(\Delta s, \Delta\mu)$ grid the time is obtained from formulae (10), (12) while the trajectory consists of nodes: $\{(i \cdot \Delta s, \Delta\mu), i = 1, \ldots, m-1\} \cup \{(0, 0), (s_m, 0)\}$ where m is the number of intervals into which the $[0, s_{\text{fin}} = s_m]$ has been divided.

(iv) $T_{sl} < T$ the robot has to stop at a suitable location on the path;

(v) $T_{sl} \ll T$ the robot has to stop at a location on the path where the rest value of the criterion is minimal.

Now we must explain what it means that the robot stops at some specific locations. According to formula (5) evaluated for $\mu = \dot{\mu} = 0$ with each discrete location into which the path was divided with step Δs, there can be associated a rest value of the criterion $J_{\text{rest}}(s)$, i.e. the value of the function under integral (9) computed for $\mu = \dot{\mu} = 0$ and $u(s) = G(s)$ (cf. Eq. (13)). Now, if $T \to \infty$, the robot has to stop temporarily at the location $(s = \hat{s})$ where the rest value is minimal and remain there to minimize the criterion. The robot rests there over a period of $T - \hat{T}$ units of time, where $\hat{T}$ denotes the time needed to move from the initial to the final location stopping at the rest location with 0 rest time while optimizing (9), obviously $\hat{T} \geq T_c$. The cost spent on the mentioned motion will be denoted as $\hat{J}$. If the location is not uniquely determined and one from among the locations which leads to the minimum rest cost corresponds to $s = 0$ or $s = s_{\text{fin}}$, it is easy to deduce that the robot will rest for $T - T_c$ units of time at the boundary location. Otherwise, all candidates for the rest location should be evaluated with respect to $\hat{J}$, and the best, with minimum $\hat{J}$, should be chosen. Generally, if T refers to the cases marked as (iii) and (iv), the robot may stop in another suitable location different than the boundary and the minimum rest-cost location. The situation, however, is not of great importance, so we assume that the only locations at which the robot is

https://doi.org/10.1017/S0263574797000647 Published online by Cambridge University Press

allowed to rest are the location of minimum rest value of the criterion or the boundary locations. Before the presentation of the algorithm steps solving the Task (***) we will introduce a shorthand notation for the input and results for one step of the Task (***).

$(\lambda, J(\lambda), T(\lambda))$ denotes that the task of minimizing the criterion (15) along the prescribed path (1), with conditions (11), (7) satisfied has been solved with parameter λ whose results are as follows: $J(\lambda)$ – the λ-independent part of the criterion described by Eq. (17) and $T(\lambda)$ – the time of the task performance.

The algorithm for solving the Task (***) runs through the following sequence of steps:

Step 1. Read the initial data T, ϵ.

Step 2. Solve the minimum time task $(\infty, J_t \stackrel{\text{def}}{=} J(\infty), T_t \stackrel{\text{def}}{=} T(\infty))$

Step 3. if $T_t > T$ then report Case (i) and STOP

Step 4. if $T - T_t < \epsilon$ then report the minimum time solution and STOP

Step 5. Solve $(0, J_c \stackrel{\text{def}}{=} J(0), T_c \stackrel{\text{def}}{=} T(0))$

Step 6. if $|T - T_c| < \epsilon$ then report the minimum cost solution and STOP

Step 7. if $T < T_c$ then solve tasks $(\lambda, J(\lambda), T(\lambda))$ with λ varied and STOP

Step 8. Determine T_{sl} as the time of the slowest admissible trajectory and $\hat{s}$, $\hat{J}$, $\hat{T}$, $J_{\text{rest}}(\hat{s})$ in the way mentioned above.

Step 9. if $T_{sl} \leq T$ then solve tasks $(\lambda, J(\lambda), T(\lambda))$ with λ varied (the result is denoted as $J_{\lambda^*} \stackrel{\text{def}}{=} J(\lambda^*)$) otherwise set $J_{\lambda^*} = \infty$ and report the case that a robot has to be stopped somewhere,

Step 10. The solution will be:

$$J^* = \min \begin{cases} (T - T_c) * J_{\text{rest}}(0) + J_c & (\#1) \\ (T - T_c) * J_{\text{rest}}(s_{\text{fin}}) + J_c & (\#2) \\ J_{\text{stop}} = (T - \hat{T}) * J_{\text{rest}}(\hat{s}) + \hat{J} & (\#3) \\ J_{\lambda^*} & (\#4) \end{cases}$$

and STOP

In Step 7 and Step 9 there have appeared tasks with λ varied. They are solved by determining an interval to which λ^* belongs and afterward shrinking the interval accordingly. For Step 7 the boundary values of λ are determined as follows: $\lambda_{\text{MIN}} = 0$ and $\lambda_{\text{MAX}} = \lambda$ such that $T(\lambda) < T$, while for Step 9 $\lambda_{\text{MAX}} = 0$ and $\lambda_{\text{MIN}} = \lambda$ such that $T(\lambda) > T$. The existence of λ meeting the above conditions is guaranteed, because $T(\infty) < T$ and $T(-\infty) > T$. When the interval has been established, the shrinking is realized by dividing the interval into halves or in the golden proportion and by choosing the subinterval to which λ^* belongs. It is not advised to assume that the time decreases linearly along the interval $[\lambda_{\text{MIN}}, \lambda_{\text{MAX}}]$.

The process of shrinking is finished when $|T(\lambda_{\text{MIN}}) - T| < \epsilon$ or $|T(\lambda_{\text{MAX}}) - T| < \epsilon$. The third condition to stop the process is $\lambda_{\text{MAX}} - \lambda_{\text{MIN}} < \epsilon_\lambda > 0$, even when the two former stop conditions are not met. This case may happen when sizes of the grid are too big or the value of ϵ is too small. It is worth noting that an additional piece of information we get while executing the algorithm is the specific form of J^* as mentioned in Step 10. So, some predictions of resulting behaviour in the case of T getting bigger can be made. For example, if any of forms #1, #2, #3 appears, the cost will increase linearly.

At the end of this section we will make some comments on the case when the prescribed time T is not strictly determined. Although the function dependence of J on T is not known, yet Properties a–d, given below, hold:

a. $T_{\text{min}} \leq T_{\text{cost}}$,
b. $J(T_{\text{min}}) \geq J(T_{\text{cost}})$,
c. $\forall T_1 \leq T_2 \in [T_{\text{min}}, T_{\text{cost}}]\, J(T_1) \geq J(T_2)$,
d. $\forall T_1 \in [T_{\text{cost}}, \infty] < T_2\, J(T_1) \leq J(T_2)$.

Properties a and b are obvious, two others need some explanations. When time traverses an interval $[T_{\text{min}}, T_{\text{cost}}]$ the cost cannot increase because the Lagrange multiplier $\lambda(>0)$ decreases and less cost consuming trajectory is chosen as a result. On the other hand, when time traverses an interval $[T_{\text{cost}}, \infty]$, the cost cannot decrease because the Lagrange multiplier $\lambda(<0)$ forces a trajectory to go through less promising areas, with respect to the cost consumption, in the phase plane to gain some time to satisfy the end-time fixed condition.

4 NUMERICAL EXAMPLES

Computer simulation tests have been carried out on the model of a PUMA-like robot depicted in Figure 1 with kinematic and dynamic parameters collected in Tables I and II, respectively. Only the first three joints (responsible for macro-motions) of the robot have been considered. Closed form equations of the dynamics of the robot have been presented in reference 12. Constraints on controls: $|u_1| \leq 40$[Nm], $|u_2| \leq 80$[Nm], $|u_3| \leq 10$[Nm].

The aim of tests was to understand the dependence of

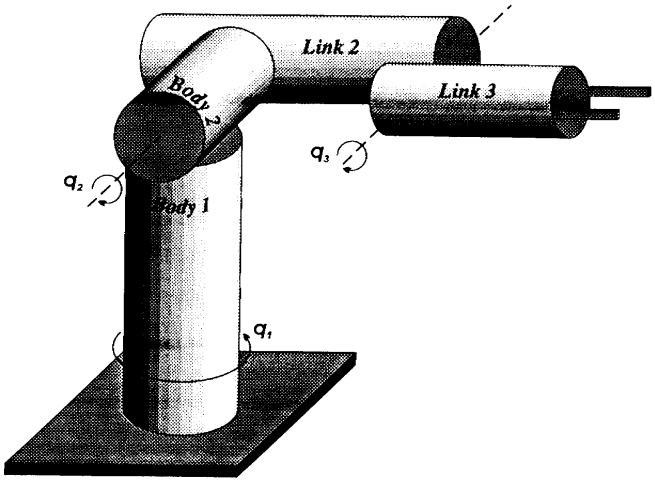


Fig. 1. Kinematic structure of the PUMA-like robot.

https://doi.org/10.1017/S0263574797000647 Published online by Cambridge University Press

Table I. Denavit-Hartenberg parameters of the PUMA-like robot

Link	q	d_i [m]	a_i [m]	α_i [rad]
1	var	0	0	$-\pi/2$
2	var	0	0.43	0
3	var	0.15	0.44	$\pi/2$

Table II. Dynamic parameters of the PUMA-like robot

Link	m [kg]	h [m]	r [m]	mass center		
				x [m]	y [m]	z [m]
1 (Body 1)	9.0	1.0	0.15	0	0.31	0.04
1 (Body 2)	4.0	0.3	0.1			
2	22.0	0.72	0.15	−0.26	0	0.4
3	5.0	0.5	0.05	−0.3	0	0

$J(u(\cdot))$ given by Eq. (17) on λ, to exemplify the solution types and to evaluate the computation spending for selected values of T. As a measure of the computation spending the number of node-to-node transitions, being the most time consuming operation, has been chosen. For testing two straight line motions (Path 1, Path 2) in the Cartesian space have been selected to exemplify various cases formulated in Step 10 of the algorithm. For the Path 2 the location of $\hat{s}$ is at the boundary of the $[0, s_{\text{fin}}]$ interval, while for the Path 1, $\hat{s}$ lies somewhere in the range.

The criterion is the minimum cost:

$$J(u(t)) = \int_0^T \sum_{i=1}^{3} |u_i(t)|\, dt. \tag{19}$$

Sizes of the grid Δs, $\Delta\mu$ are constant and equal to $s_{\text{fin}}/32$ and 50 [mm/s], respectively. The parameters used in the stop conditions checking are as follows: $\epsilon = 0.05$ [s], $\epsilon_\lambda = 0.05$.

At first the functions: $J(\lambda)$, $T(\lambda)$, $J(T)$, have been determined. Their plots presented in Figures 2, 3, 4 satisfy Properties a–d formulated in Section 3.

The figures presented clearly confirm that the function $T(\lambda)$ is not increasing. Another observation is that constant sizes of the grid can cause that the plot of $T(\lambda)$ might not be smooth, so sizes of the grid should be chosen carefully – much finer when μ is small; for more instructions concerned with the grid sizes see reference 6.

For the cost-optimal fixed-time task along Path 1 and with $T = 1.22$ [s] joint coordinates and controls have been presented in Figures 5 and 6, respectively. Controls at the beginning/the end of motion have relatively large values in order to increase/decrease velocity along the path from/to zero. Cost-optimal trajectory planning forces to reach quickly a region in the phase-plane with small cost spending.

Table III sums up representative results for specific subtasks of the Task (***) (minimum-time task, minimum cost task with free right-end time, etc.).

The meaning of most entries of Table III has been defined previously; few new terms have the following meaning: λ_{up} is the minimum value of λ for which the solution in the phase-plane looks like the minimum-time one; λ_{sl} is the maximum value of λ for which the solution looks like the slowest one under condition that touching s-axis except at boundary points is not allowed; λ_{switch}, T_{switch}, J_{switch} – values of λ, T, J at which the type of solution switches between various types: varying λ (type #4) and resting at the boundary point or at the location where the rest criterion achieves minimum. Table IV shows the computation spending (the number node-to-node transitions) for selected tasks (in brackets the following numbers appear: time prescribed, time computed, value of criterion, value of λ^*, the number of time-free tasks solved until the time-fixed task have been completed). The tasks with right-end time free have been solved by the method of minimal neighbourhood.[8] As can be seen from Table IV approximately a dozen of free-time tasks have to be solved to reach a result of a fixed-time task. For comparison, the last row of Table IV contains the computaton speending for only one free-time task solved by the dynamical programming method. The computation spending for the time-fixed task (when the method of minimal neighbourhood[8] is applied) is, even in the worst case, significantly less than the computation spending for a single task solved by DP method. All remarks made are not influenced by the type of paths considered.

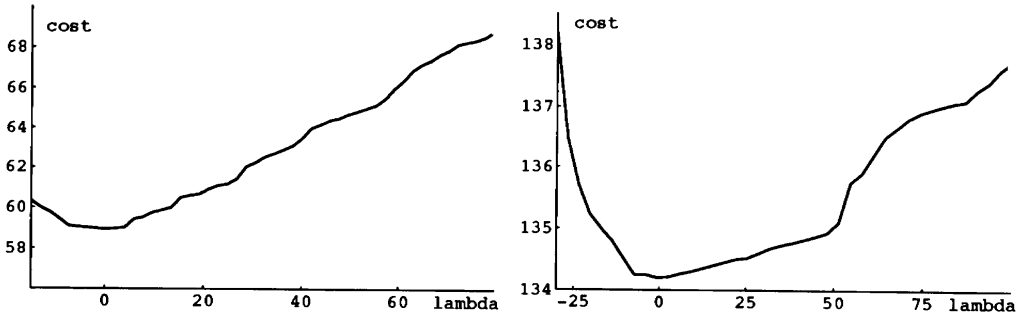


Fig. 2. λ-independent part of the criterion, cf. formula (17), as a function of λ.

https://doi.org/10.1017/S0263574797000647 Published online by Cambridge University Press

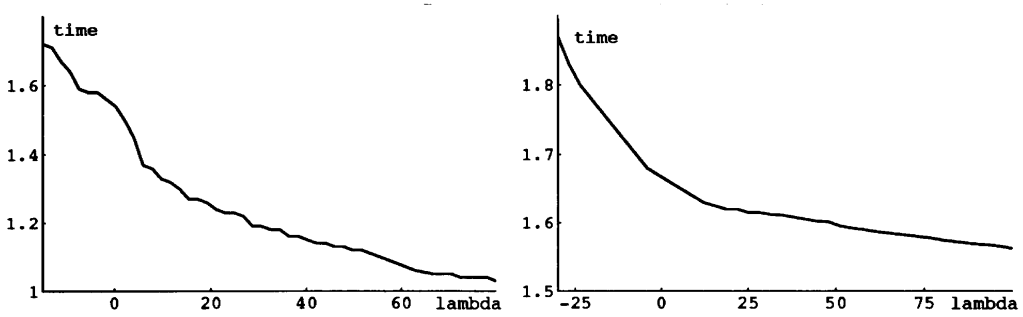


Fig. 3. Dependence of $T(\lambda)$ on λ.

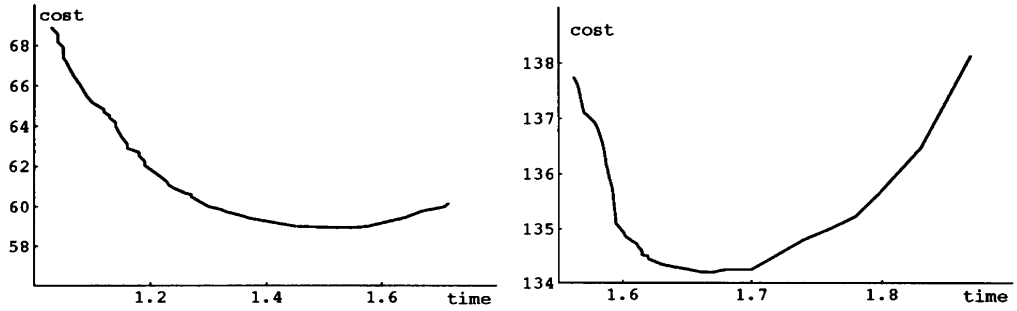


Fig. 4. Minimum cost solution as a function of prescribed T.

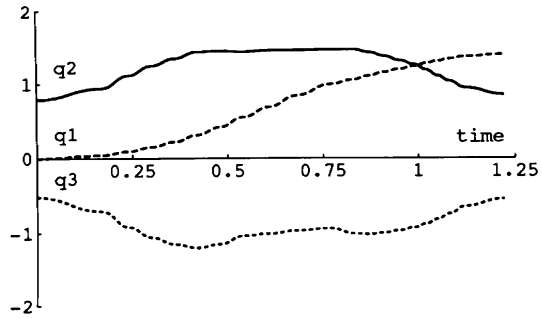


Fig. 5. Fixed-time ($T = 1.22[s]$) cost-optimal trajectory planning along Path 1 – joint coordinates.

Table III. Representative results for Task (***) – summary

Parameter	Task 1	Task 2
s_{fin} [mm]	1246.4	1264
T_t [s]	0.945	1.54
J_t [−]	81.88	141.45
λ_{up}	300	494
T_c [s]	1.54	1.67
J_c [−]	58.9	134.2
$\hat{s}$ [mm]	831.3	0
$\hat{T}$ [s]	1.72	1.67
$\hat{J}$ [−]	66.95	134.2
$J_{\text{rest}}(\hat{s})$ [−]	17.34	57.2
T_{sl} [s]	27.3	26.75
J_{sl} [s]	830.5	1590
λ_{sl}	−50	−65
λ_{switch}	−19	−57
T_{switch} [s]	2.5	8.8
J_{switch} [−]	80.4	542.0

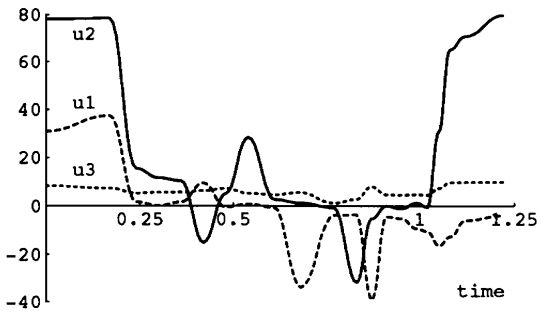


Fig. 6. Fixed-time ($T = 1.22[s]$) cost-optimal trajectory planning along Path 1 – controls.

Table IV. Computation spending for selected values of T

Task 1	*Task* 2
10 131 (1.2, 1.206, 61.72, 27.63, 16)	8 877 (1.6, 1.603, 134.9, 41.9, 7)
10 628 (1.8, 1.804, 61.77, −17.96, 11)	12 762 (1.8, 1.805, 136.3, −25.45, 18)
22 967	13 730

https://doi.org/10.1017/S0263574797000647 Published online by Cambridge University Press

5. APPLICATIONS OF THE TIME-MATCHING METHOD

With the help of the time-matching method and exploiting Properties a–d from Section 3 the following task can be solved:

Task 1: For initial data as in the Task (*) with more time-flexible condition $T_{\text{low}} \le T \le T_{\text{up}}$ for T_{low}, T_{up} constant, find a cost optimal solution.

The solution is described by one of the following cases; ($T_{\min}, T_{\text{cost}}, J_{\text{cost}}$ have been defined in Section 3)

1. if $T_{\text{up}} < T_{\min}$, there is no solution,
2. if $T_{\text{low}} \le T_{\text{cost}} \le T_{\text{up}}$, the solution is J_{cost}
3. if $T_{\text{up}} < T_{\text{cost}}$, the solution is $J(T_{\text{up}})$
4. if $T_{\text{low}} > T_{\text{cost}}$, the solution is $J(T_{\text{low}})$.

In cases 3 and 4 the right-end fixed-time task must be solved (most effectively by the time-matching method presented in Section 3).

Another application of the time-matching method is in

Task 2: an optimization of a simultaneous motion of two robots following a prescribed, non-colliding paths, when both of them must be ready simultaneously for their next tasks. In this case criterion functions become:

$$J_1(u_1) = \int_0^T L_1(q_1, \dot{q}_1, u_1)\, \mathrm{d}t,$$

$$J_2(u_2) = \int_0^T L_2(q_2, \dot{q}_2, u_2)\, dt.$$

and the optimal controls u_1^*, u_2^* must satisfy

$$J_1(u_1^*) + J_2(u_2^*) = \min_{u_1, u_2} (J_1(u_1) + J_2(u_2)),$$

with the time T free, yet equal for both robots.

A computationally expensive solution of Task 2 is to vary T, to solve (for each particular T) fixed right-end time tasks for both robots, to collect a sum of costs, and to take as the solution its minimal value. Even in this case an advantage of the particular shape of $J(T)$ helps considerably to simplify the Task 2. As it can be easily proven[13] the solution must be for time T somewhere in the interval $[\max(\min\{T^1_{\text{cost}}, T^2_{\text{cost}}\}, \max\{T^1_{\min}, T^2_{\min}\}), \max\{T^1_{\text{cost}}, T^2_{\text{cost}}\}]$, where superscripts denote the number of the robot. Although the interval has been shrunk, nevertheless the time matching method must be applied to traverse the interval. Mixing Tasks 1 and 2 we get another task demanding to apply the time-matching method.

The approach can be generalized for more than two robots following non-colliding paths. Especially in the multi-robot systems an efficiency of solving the time-fixed task is of a primary importance, because a lot of such tasks must be solved to get a final result.

6. CONCLUSIONS

In this paper a method and an algorithm for suboptimal-cost trajectory planning with fixed time are presented. The efficiency of the algorithm based on extensive use of the method of minimal neighbourhood (used as a tool in solving elementary free-time subtasks) has been confirmed by simulation tests. The condition of fixed time has been weakened to cover more time-flexible cases. These cases (e.g. a time interval prescribed) are governed by Properties a–d given in Section 3. Applications of the algorithm may be relevant to optimizing multi-robot systems as it has been shown in Section 5 or to a task planning in robot-based technological lines. The experiments have pointed out that it is important not only to plan the motion of the robot but also the location at which the robot should remain at rest. This paper attempted not only to state, but also to effectively solve some tasks arising at the boundary between single robot and multi-robot systems.

Acknowledgments

The author wish to express his gratitude to Prof. Krzysztof Tchoń for his help and support and to an anonymous reviewer for helpful comments.

References

1. K.G. Shin and N.D. McKay, "Minimum-time control of robotic manipulators with geometric path constraints" *IEEE Transactions* **AC-30,** No. 6, 531–541 (1985).
2. B.K. Kim and K.G. Shin, "Minimum-time path planning for robot arms and their dynamics" *IEEE Transactions* **SMC-15,** No. 2, 213–223 (1985).
3. H. Ozaki and A. Mohri, "Synthesis of a minimum-time manipulator trajectories with path constraints using time scaling" *Robotica* **6,** Part 1 41–46 (1988).
4. F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning", *IEEE J. Robotics and Automation,* **3,** No. 2, 115–123 (1987).
5. K.G. Shin and N.D. McKay, "A dynamic programming approach to trajectory planning of robotic manipulators" *IEEE Transactions* **AC-31,** No. 6, 491–500, (1986).
6. K.G. Shin and N.D. McKay, "Minimum cost trajectory planning for industrial robots" **In:** *Advances in Robotic Control Systems,* (ed. by M.C. Leondes), (Academic Press, New York, 1991) pp. 357–403.
7. W. Jacak, I. Dulęba and P. Rogaliński, "A graph searching approach to trajectory planning of robot's manipulator". *Robotica* **10,** part 6, 531–537 (1992).
8. I. Dulęba, "The method of minimal neighbourhood: a new and most effective iterative method for trajectory planning in robot manipulators" *Robotica* **13,** part 3, 297–304 (1995).
9. M. Vukobratović and M. Kirćansky, "A method of optimal synthesis of manipulator robot trajectories", *Transactions* **ASME-104,** 188–193 (1982).
10. I.M. Gelfand and S.W. Fomin, "Calculus of variations", *PWN* (In Polish; translated from Russian) (1975).
11. R.P. Paul, *Robot Manipulators: Mathematics, Programming and Control* (MIT Press, Cambridge, MA, 1981).
12. B. Buchberger et al., 'Hybrid Programming Approach to Robot Dynamics" *Technical Report* (RWC project, Research Institute for Symbolic Computation, Linz, 1994).
13. I. Dulęba, "Cost optimal trajectory planning with right-end time fixed", *Int. Symp. on Intelligent Robotic Systems SIRS,* Lisbon, Portugal, pp. 73–78 (1996).

https://doi.org/10.1017/S0263574797000647 Published online by Cambridge University Press