A Hierarchical Decomposition for Large-scale Optimal Control Problems with Parallel Processing Structure*

SHI-CHUNG CHANG,† TSU-SHUAN CHANG‡ and PETER B. LUH§

A hierarchical, time decomposition and coordination scheme for long horizon optimal control problems is suitable for parallel processing and adds a new dimension to results on large-scale dynamic optimization.

Key Words—Dynamic optimization; decomposition and coordination; hierarchical optimization; parallel processing.

Abstract—This paper presents a new method in solving long horizon optimal control problems. The original problem is decomposed along the time axis to form many smaller subproblems, and a high level problem is created that uses initial and terminal states of subproblems as coordination parameters. In such a scheme, the high level problem is a parameter optimization problem. Subproblems are optimal control problems having shorter time horizon, and are completely decoupled so that they can be solved in parallel. It is shown that the two-level problem has the same global optimum as the original one. Moreover, the high level problem is a convex programming problem if the original problem has a convex cost function and linear system dynamics. A parallel, two-level optimization algorithm is then presented, where the the high level problem is solved by Newton's method, and low level subproblems are solved by the Differential Dynamic Programming technique. Numerical testings on two examples are given to illustrate the idea, and to demonstrate the potential of the new method in solving long horizon problems under a parallel processing environment.

1. INTRODUCTION

There are many computational methods for solving optimal control problems. According to the nature of results, these methods can be categorized into three classes: open-loop, feedback, and closed-loop (Polak, 1973). For example, methods based on dynamic programming yield closed-loop solutions while the use of maximum principle results in open-loop solutions. As closed-loop solutions are generally difficult to obtain and pure open-loop solutions are not satisfactory for practical applications, methods for obtaining solutions with certain feedback properties are typically adopted (Findeisen et al., 1980). Though there are many existing techniques for solutions of different nature, there have not been many efficient methodologies for solving large-scale problems. As the computational power increases, such as the availability of faster hardwares and cost-effective parallel processors, etc., the size and scope of problems we want to tackle also grow. Enormous amount of research interests have lately been invoked to seek for efficient solution methodologies for large-scale problems by exploiting both problem structure and advanced computing techniques, especially the parallelization of algorithms for the application of parallel processing systems.

One popular scheme in handling large-scale optimization problems, either static or dynamic, is decomposition and coordination: a large problem is decomposed, based on problem structure, into small subproblems which can be solved efficiently, and a proper coordination scheme is created to glue subproblems together and to insure the optimality of the solution. Parallelism is usually achieved as a result of decomposition. Methods such as decomposition by pricing mechanism, decomposition by right-hand-side allocation, the generalized Benders' decomposition, etc. (Lasdon, 1970; Geoffrion, 1972; Silverman, 1972; Shapiro, 1979) have been developed in the mathematical programming literature for static problems. Some of them have been applied to dynamic problems by treating systems dynamics as structured constraints and then adopting a static viewpoint.

* Received 22 October 1986; revised 29 June 1987; revised 24 May 1988; received in final form 1 June 1988. The original version of this paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor M. Jamshidi under the direction of Editor A. P. Sage.

† Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06268, U.S.A., now at Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, Republic of China.

‡ Department of Electrical and Computer Engineering, University of California, Davis, CA 95616, U.S.A.

§ Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06268, U.S.A.
A high level parameter optimization problem is chosen as coordination parameters. For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).

For many large-scale optimal control problems, the long time horizon adds another dimension of difficulty. In this paper, we present a scheme that decomposes a long horizon problem into smaller subproblems along the time axis. The initial and terminal states of subproblems are chosen as coordination parameters. A parallel dynamic programming algorithm based on state variable decomposition was presented in Scheel (1981).
Communications take a negligible amount of time. For loosely coupled systems where interprocessor communication times cannot be ignored, our measures provide bounds on actual performances.

The paper is organized as follows. The two-level optimization problem is formulated in Section 2. It is shown in Section 3 that the two-level problem has the same global optimum as the original problem. A sufficient condition for the high level problem to be a convex programming problem is also given. In Section 4, the two-level, parallel NM-DDP algorithm is presented. Section 5 provides numerical testing results on two examples. Concluding remarks are then given in Section 6.

2. THE TWO-LEVEL FORMULATION

Consider the following optimal control problem:

\[
\begin{align*}
\text{(P)}: \min_{u_i} J, \quad & \text{with } J = g_N(x_N) + \sum_{i=0}^{N-1} g_i(x_i, u_i), \\
\text{subject to } & \text{the system dynamics } \quad x_{i+1} = f_i(x_i, u_i), \quad i = 0, \ldots, N - 1, \quad \text{with } x_0 \text{ given,} \\
\text{and constraints } & \quad x \in X \subset R^n, \quad \text{and } u \in U \subset R^m. 
\end{align*}
\]

(2.1)

Assume that problem (P) has a finite optimal cost and that \( N = MT \gg 1 \), where \( M \) and \( T \) are both positive integers. By specifying the values of \( \{x_{(j-1)T}\}_j \), we can decompose (P) into \( M \) independent, \( T \)-stage subproblems as follows.

\[
\begin{align*}
\text{(P-j) Subproblem } j, \quad & \text{with } j = 1, \ldots, M \\
\min_{u_k} J_j, \quad & \text{with } J_j = \sum_{k=(j-1)T}^{jT-1} g_k(x_k, u_k), \\
\text{subject to } & \text{the system dynamics } \quad x_{k+1} = f_k(x_k, u_k), \quad (j-1)T \leq k \leq jT - 1, \\
\text{and the corresponding constraints of (2.3).}
\end{align*}
\]

(2.4)

The above scheme and the one presented in Chang and Luh (1985) are both time decomposition schemes. The major difference is that, instead of modifying subproblems' cost functions as a means of coordination, we choose initial and terminal states of subproblems as coordination terms.

3. ACHIEVABILITY OF OPTIMAL SOLUTIONS

In this section, we shall show that problems (P) and (P-H) have the same global optimum. We shall also show that the high level problem (P-H) is a convex programming problem if the original cost function \( g_i \) is convex, the admissible state space \( X \) and the admissible control space \( U \) are convex, and the system dynamics \( f_i \) is linear.

Theorem 1. Problems (P-H) and (P) have the same global optimum.

To prove Theorem 1, we first convert problem (P) into a parameter optimization problem. We then utilize the fact that a parameter minimization problem can be converted into a two-level minimization problem by first minimizing over a subset of parameters and then minimizing over the remaining parameters. Details are given in Appendix A.

From Theorem 1, we see that instead of solving the original long horizon optimal control problem (P), we can solve a two-level problem. The high level problem (P-H) is a parameter optimization problem. Low level subproblems (P-j)s are optimal control problems of a shorter time horizon. These subproblems can be solved in parallel, as they are decoupled once \( \{x_{jT}\}_j \) is set by (P-H).

Though Theorem 1 guarantees that both (P) and (P-H) have the same global optimum, it is not clear whether the two-level formulation changes or creates local optima. In the following theorem, we present a sufficient condition for (P-H) to be a convex parameter optimization problem and therefore have the same optimal as (P).

Theorem 2. Problem (P-H) is a convex para-
meter optimization problem over the set \( X_H \) if
(i) \( g_i(x_i, u_i) \) is convex,
(ii) \( f(x_i, u_i) \) is linear, and
(iii) \( X \) and \( U \) in (2.3) are convex sets.

Proof. Since the cost function in (P-H) is of the additive form as given by (2.6), to prove the theorem, it is sufficient to show that \( J^*_\alpha(x_{0:T}, x_T) \) is convex in terms of its arguments. We therefore consider subproblem (P-j). For simplicity of presentation, the subproblem index \( j \) is omitted and \( x(i_\sim) \in X \) is written as \( x_0 \) in the following proof. Define the set of admissible state and control trajectories \( XUA \) as

\[
XUA = \{(x_N, \ldots, x_1, u_{N-1}, \ldots, u_0) \in X^N \times U^N \mid (2.2) \text{ holds for } (x_N, \ldots, u_0)\}. \tag{3.1a}
\]

Then define the set of admissible initial and terminal states \( X_{0:T} \) as

\[
X_{0:T} = \{(x_0, x_T) \mid \text{ there exists a trajectory } (x_T, \ldots, x_1, u_{T-1}, \ldots, u_0) \in XUA\} \tag{3.1b}
\]

The proof is then reduced to the proof of lemma 1.

Lemma 1. (a) \( X_{0:T} \) is a convex set. (b) \( J^*(x_0, x_T) \) is convex over \( X_{0:T} \).

Proof. (a) Since \( f_i(x_i, u_i) \) is linear, we let

\[
x_{i+1} = A_i x_i + B_i u_i, \quad i = 0, \ldots, T - 1, \tag{3.2}
\]

where \( A_i \in R^{m_i} \) and \( B_i \in R^{m_i} \). From (3.2), we obtain

\[
x_0 = \sum_{i=0}^{T-1} \Phi_i B_i u_i, \tag{3.3}
\]

where

\[
\Phi_i = \prod_{j=i+1}^{T-1} A_j \quad \text{and} \quad \Phi_{T-1} = I.
\]

Let \( x_{i_\sim}^0, x_{i_\sim}^1 \) and \( (u_{i_\sim})^0, (u_{i_\sim})^1, i = 0, \ldots, T - 1, k = 1, 2 \), denote two sets of quantities satisfying (3.3). Define

\[
x_0^0 = \alpha x_0^1 + (1 - \alpha)x_0^2, \tag{3.4a}
\]

\[
x_T^0 = \alpha x_T^1 + (1 - \alpha)x_T^2, \tag{3.4b}
\]

\[
u_i^0 = \alpha u_i^1 + (1 - \alpha)u_i^2, \quad i = 0, \ldots, T - 1, \tag{3.4c}
\]

for an arbitrary \( \alpha \in [0, 1] \). Note that \( x_0^0 \in X \), \( x_T^0 \in X \) and \( u_i^0 \in U \) as \( X \) and \( U \) are convex sets. We then have

\[
x_0^0 = \Phi_{-1}(\alpha x_0^0 + (1 - \alpha)x_0^0) + \sum_{i=0}^{T-1} \Phi_i B_i (\alpha u_i^0 + (1 - \alpha)u_i^0), \tag{3.4d}
\]

which implies \( (x_0^0, x_T^0) \in X_{0:T} \). Since this is true for an arbitrary \( \alpha \in [0, 1] \), we conclude that \( X_{0:T} \) is a convex set.

(b) For a given pair of \( (x_0^k, x_T^k) \), \( k = 0, 1, 2 \) as defined in part (a), denote \( (x_0^k, u_0^k) \) as its optimal state and control trajectories and \( J^*(x_0^k, x_T^k) \) its corresponding cost. To prove that \( J^*(x_0, x_T) \) is convex in its arguments, we have to show that

\[
J^*(x_0^0, x_T^0) \leq \alpha J^*(x_0^1, x_T^1) + (1 - \alpha) J^*(x_0^2, x_T^2). \tag{3.5}
\]

Define

\[
U(x_0, x_T) = \{(u_0, i = 0, \ldots, T - 1) \in U^T \mid (14) \quad \text{holds for the given } x_0 \text{ and } x_T\}. \tag{3.6}
\]

Note that \( X \) and \( U \) are convex sets. By letting

\[
\bar{x}_i = \alpha x_i^1 + (1 - \alpha)x_i^2, \quad \text{and} \quad \bar{u}_i = \alpha u_i^1 + (1 - \alpha)u_i^2, \tag{3.7}
\]

we have \( (\bar{u}_i, i = 0, \ldots, T - 1) \in U(x_0^0, x_T^0) \) and \( (\bar{x}_i) \) is the state trajectory for the control sequence \( (\bar{u}_i) \). Therefore,

\[
J^*(x_0^0, x_T^0) = \min_{(u_0, \ldots, u_{T-1}) \in U(x_0^0, x_T^0)} g_T(x_T) + \sum_{i=0}^{T-1} g_i(x_i, u_i) \tag{3.8}
\]

\[
\leq \sum_{i=0}^{T-1} g_i(x_i^1, u_i^1) \tag{3.8}
\]

\[
+ (1 - \alpha) \sum_{i=0}^{T-1} (g_i(x_i^2, u_i^2)) = \alpha J^*(x_0^1, x_T^1) + (1 - \alpha) J^*(x_0^2, x_T^2).
\]

The first inequality holds because of (i) above, and the second comes from the convexity of \( g_i \).

The proof is thus completed.

4. A PARALLEL PROCESSING ALGORITHM

As formulated in the previous section, the high level problem (P-H) is a parameter optimization problem, and many methods for parameter optimization can be used to solve it. Similarly, many optimal control techniques can be used to solve low level (P-j)s. The selection and integration of methods to solve the two-level problem should be carefully made based on desired nature of results and overall computational efficiency. We present in this section a parallel, two-level algorithm to solve problems without state/control variable constraints, i.e. \( x_i \in X = R^m \) and \( u_i \in U = R^m \). Our emphases are to illustrate the idea of the two-level approach, and to demonstrate its potential for parallel processing of long horizon optimal control problems.

We assume in this section that all functions in
(P) are smooth so that all needed derivatives exist, and that the system is T-stage controllable, i.e. for any given pair of initial and terminal states of subproblem (P-j), there is a feasible solution. Since we are dealing with long horizon problems, T in principle is large. Furthermore, there is no constraint on control variables. Therefore T-stage controllability is not a stringent assumption. Newton's method is selected for solving (P-H) as follows:

\[ x_{k+1}^k = x_k^k - [\nabla^2 J(x_k^k)]^{-1} \nabla J(x_k^k), \]  

(4.1)

where \( x_k = (x_{1,T}, \ldots, x_{MT})' \) and \( k \) is the iteration index.

From (2.6), we have

\[ \frac{\partial J}{\partial x_{JT}} = \frac{\partial J^*_j}{\partial x_{JT}}(x_{(j-1)T}, x_{JT}) + \frac{\partial J^*_j}{\partial x_{JT}}(x_{(j+1)T}, x_{(j+1)T}) \]  

(4.2)

Similarly, the Hessian \( \nabla^2 J \) consists of Hessians of \( J_j(x_{(j-1)T}, x_{JT}), j = 1, \ldots, M, \) and is an M-level block tridiagonal matrix:

\[ \nabla^2 J = \begin{bmatrix}
  b_1 & c_1 & 0 & & \\
  a_2 & b_2 & c_2 & & \\
  & a_3 & b_3 & c_3 & \\
  & & \ddots & \ddots & \\
  & & & b_{M-1} & \\
  & & & & b_M
\end{bmatrix}, \]  

(4.3)

where \( b_j = \partial^2(J^*_j + J^*_{j+1})/\partial x_{JT}^2, a_j = \partial^2(J^*_{j-1} + J^*_j)/\partial x_{(j-1)T}^2 \partial x_{JT}, \) and \( c_j = \partial^2(J^*_{j+1} + J^*_{j+2})/\partial x_{JT}^2 \partial x_{(j+1)T}. \) Note that both \( \nabla J \) and \( \nabla^2 J \) can be computed locally by the jth subproblem and passed onto the high level to form \( \nabla J \) and \( \nabla^2 J. \) The problem then reduces to how to find the needed information efficiently while solving low level subproblems. Based on this concern, an extended Differential Dynamic Programming (DDP) technique is adopted at the low level.

The DDP is a successive approximation technique for solving optimal control problems with free terminal states (Jacobson and Mayne, 1970; Ohno, 1978; Yakowitz and Rutherford, 1984; Yakowitz, 1986). It has been extended to problems with fixed terminal states in Chang et al. (1986a), and Chang (1986). It consists of two procedures: backward dynamic programming and successive policy construction. For a low-level subproblem (P-j), a backward dynamic programming procedure is applied by taking a quadratic approximation of (P-j) along a nominal trajectory, and formulating at each stage a quadratic programming problem in variational terms of control and state variables.

By solving the quadratic programming problem at each stage, coefficients of the linear optimal variational control and coefficients of the quadratic variational cost-to-go function are obtained. The successive policy construction procedure uses these control coefficients and the nominal controls to construct new controls forward in time, and to calculate the new cost. If the cost is lower than the nominal one, the nominal trajectory is updated by the new trajectory. Otherwise the new control is modified in a specific way till the constructed control yields a cost lower than the nominal one. These forward and backward procedures are carried out repetitively to obtain a convergent solution. The information required by the high level Newton method is readily available from coefficients of the variational cost-to-go function at convergence. A more detailed description of DDP is given in Appendix B. The two-level optimization algorithm with parallel low-level subproblems is depicted in Fig. 1. Note that in the algorithm implemented, the matrix \( B_{T-1} \) is assumed to be invertible to simplify the handling of constraints caused by the fixed terminal state. Analysis for the case where \( B_{T-1} \) is not invertible can be found in Chang (1986).

To further enhance the parallelism of the algorithm, the block tridiagonal structure of \( \nabla^2 J \) is exploited and the cyclic odd–even reduction algorithm of Heller (1976) is adopted to find the Newton step

\[ y = [\nabla^2 J(x_n)]^{-1} \nabla J(x_n), \]  

(4.4)

at the high level optimization. This algorithm is a parallel algorithm that solves the block tridiagonal equation

\[ [\nabla^2 J(x_n)] y = \nabla J(x_n). \]  

(4.5)

To briefly explain the idea of cyclic reduction, let us rewrite (4.5) in block terms defined in (4.3). The jth equation becomes

\[ a_j y_{j-1} + b_j y_j + c_j y_{j+1} = v_j, \]  

(4.6)

where \( v_j \) is a vector consisting of the components of \( \nabla J \) corresponding to the jth block. Assume for simplicity that \( M = 2^m - 1 \), where \( m \) is a positive integer. There are two basic operations in cyclic reduction: reduction and back substitution. If we multiply equation \( 2j-1 \) by \(-a_{2j}b_{2j-1}^{-1}\), equation \( 2j+1 \) by \(-c_{2j}b_{2j+1}^{-1}\), and add them to equation \( 2j \), we can eliminate the odd-indexed, off-diagonal blocks. Picking up blocks associated with the even-indexed unknowns, we form again a tridiagonal equation with \( 2^m - 1 \) levels of blocks. The procedure is repeated till a one-block equation is obtained. This is the
High level
Compute \( J, \psi_J, \) and \( \frac{\partial J}{\partial x_i} \).
Update \( x, t = 1, \ldots, M \) by
Newton's Method (eq. 21).

Subproblem \( k \)
DDP

\[ x_k \text{ given} \]

Subproblem \( k \)
DDP

\[ x_k \text{ given} \]

Subproblem \( k \)
DDP

Subproblem \( M \)
DDP

\[ x_M \text{ given} \]

Fig. 1. The Newton-DDP algorithm with parallel structure.

5. NUMERICAL RESULTS

As pointed out in Yakowitz and Rutherford (1984), there is no well-established, standard testing problem for large-scale optimal control systems. In this section, two problems with nonquadratic objective functions and linear system dynamics are adopted for the testing purpose. The one level DDP with free terminal state is also tested as a basis for comparison.

As mentioned, numerical testings are performed in single precision on an IBM 3084 mainframe computer due to a lack of a parallel processor. A user supplied subroutine (provided by UConn Computer Center) is used to time the execution of the algorithm. Two performance measures, speedup and efficiency (to be explained later) are obtained assuming no communication cost among processors. These performance measures are good for tightly coupled parallel processing systems where memory is shared and interprocessor communications take a negligible amount of time. For loosely coupled systems where interprocessor communication times cannot be ignored, our measures provide bounds on actual performances. The timing results listed throughout this section are in units of seconds. Testing results demonstrate the feasibility of the decomposition approach, and the advantage of the algorithm for solving long horizon problems under a parallel processing environment.

Example 1. Cost function:

\[
J = \sum_{i=1}^{m} \left[ \sum_{j=1}^{n} (x_{ij} - a_{ij})^2 \sum_{i=1}^{m} u_{ii}^2 + \sum_{i=1}^{m} u_{ii}^2 \right.
\]

\[ + \sum_{i=1}^{m} \sum_{j=i}^{m} u_{ij} u_{ji} + 100 \sum_{i=1}^{m} (x_{ii} - a_{ii})^2 \right].
\]

(5.1)

where \( m = n = 4 \), and \( \{a_{ii}\} \) is an independent, identically distributed random sequence with uniform distribution over the interval \([-2, 2]\).

This cost function is designed to let \( x_{ii} \) close to \( a_{ii} \) and have small controls. It is convex when magnitudes of \( x \) and \( u \) are not too large.

System dynamics:

\[
A = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\]

and \( B = \begin{bmatrix}
-1 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 \\
0 & 1 & -1 & 0 \\
0 & 0 & 1 & -1
\end{bmatrix} \). (5.2)
Time decomposition for optimal control

### TABLE 1. TEST RESULTS FOR TWO PERFORMANCE MEASURES

<table>
<thead>
<tr>
<th>Stages</th>
<th>( T_1 )</th>
<th>( M \times T )</th>
<th>( L_p )</th>
<th>( S_p )</th>
<th>( E_p )</th>
<th>( f.c. )</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>0.59</td>
<td>0.1764597</td>
<td>0.36</td>
<td>1.639</td>
<td>0.546</td>
<td>0.1764564</td>
</tr>
<tr>
<td>84</td>
<td>1.12</td>
<td>0.3430279</td>
<td>0.61</td>
<td>1.836</td>
<td>0.612</td>
<td>0.3430364</td>
</tr>
<tr>
<td>168</td>
<td>2.21</td>
<td>0.7067726</td>
<td>1.12</td>
<td>1.973</td>
<td>0.658</td>
<td>0.7068281</td>
</tr>
<tr>
<td>336</td>
<td>4.41</td>
<td>1.240006</td>
<td>2.15</td>
<td>2.051</td>
<td>0.684</td>
<td>1.240177</td>
</tr>
<tr>
<td>42</td>
<td>7 x 6</td>
<td>0.31</td>
<td>1.90</td>
<td>0.271</td>
<td>0.1764609</td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>7 x 12</td>
<td>0.43</td>
<td>2.605</td>
<td>0.372</td>
<td>0.3430302</td>
<td></td>
</tr>
<tr>
<td>168</td>
<td>7 x 24</td>
<td>0.73</td>
<td>3.027</td>
<td>0.432</td>
<td>0.7068199</td>
<td></td>
</tr>
<tr>
<td>336</td>
<td>7 x 48</td>
<td>1.16</td>
<td>3.80</td>
<td>0.543</td>
<td>1.240194</td>
<td></td>
</tr>
</tbody>
</table>

f.c.: final cost/10^4?

The initial state is \( x_0 = [0 0 0 0]' \), and the initial nominal control is \( u_t = 0 \) for all \( i \) and \( t \).

### Convergence criteria

The convergence criteria for the DDP algorithm is of the following form:

\[
|J^k_{t+1} - J^k_t|/J^k_t \leq \eta. \tag{5.3}
\]

The convergence threshold \( \eta = 0.0001 \) for the one-level DDP, and \( \eta = 0.001 \) for the DDP in the two-level algorithm. For the high level optimization, the criterion is either (5.3) with threshold \( \eta = 0.0001 \) or \(|\nabla J| < 0.001\).

To measure the performance of the algorithm, let \( T_1 \) be the running time of the one-level DDP, \( T_2 \) be the total sequential running time of the two-level algorithm, \( T_p \) be the corresponding low level parallel running time (to be discussed later), and \( M \) be the number of processors which is assumed to be equal to the number of subproblems. Communication time is ignored by assuming that either a shared memory is available, or communication time is relatively small. At the \( k \)th iteration, let \( t_k \) denote the longest processing time among all processors in solving low level subproblems. The low level parallel running time \( T_p \) is calculated by adding up all the \( t_k \)'s. The high level computation is assumed to be done by \( M \) processors in parallel, and its computation time \( T_h \) is approximated by \((T_2 - T_p)/M\). This approximation is justified by testing results that \( T_2 - T_p \ll T_p \) (approximately \( 1:15 \)), as the high level Newton optimization is much simpler than the low level DDP. Two performance measures are then adopted (Heller, 1978). Speedup, defined as \( S_p = T_1/(T_1 + T_h) \), measures improvement in computation time by using parallel processing. On the other hand, efficiency \( E_p = S_p/M \), measures how well the processing power is utilized. Testing results for a sequence of \( \{a_n\} \) are given in Table 1.

### TABLE 2. STATISTICS FOR THE TWO PERFORMANCE MEASURES OF EXAMPLE 1 BASED ON MONTE CARLO SIMULATION

<table>
<thead>
<tr>
<th>( M )</th>
<th>( M \times T )</th>
<th>( \delta_p )</th>
<th>( \sigma_p^2 )</th>
<th>( \epsilon_p )</th>
<th>( \epsilon_p^2 )</th>
<th>( A_e )</th>
<th>( \epsilon_e^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>42</td>
<td>1.35</td>
<td>0.293</td>
<td>0.448</td>
<td>3.25</td>
<td>0.325</td>
<td>0.044</td>
</tr>
<tr>
<td>3</td>
<td>84</td>
<td>1.37</td>
<td>0.049</td>
<td>0.455</td>
<td>0.551</td>
<td>0.463</td>
<td>0.037</td>
</tr>
<tr>
<td>3</td>
<td>168</td>
<td>1.54</td>
<td>0.074</td>
<td>0.513</td>
<td>0.819</td>
<td>0.638</td>
<td>0.189</td>
</tr>
<tr>
<td>3</td>
<td>336</td>
<td>1.56</td>
<td>0.052</td>
<td>0.519</td>
<td>0.579</td>
<td>0.005</td>
<td>0.193</td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>2.21</td>
<td>0.942</td>
<td>0.315</td>
<td>1.92</td>
<td>0.854</td>
<td>0.109</td>
</tr>
<tr>
<td>7</td>
<td>84</td>
<td>2.70</td>
<td>0.143</td>
<td>0.385</td>
<td>0.292</td>
<td>1.32</td>
<td>0.431</td>
</tr>
<tr>
<td>7</td>
<td>168</td>
<td>3.12</td>
<td>0.311</td>
<td>0.446</td>
<td>0.635</td>
<td>0.010</td>
<td>0.041</td>
</tr>
<tr>
<td>7</td>
<td>336</td>
<td>3.26</td>
<td>0.118</td>
<td>0.465</td>
<td>0.242</td>
<td>0.178</td>
<td>0.052</td>
</tr>
</tbody>
</table>

\( \delta_p \): mean of \( S_p \), \( \sigma^2 \): variance.

\( A_e \): final cost accuracy (percentage difference between one- and two-level final costs relative to the one-level final cost).

To further study the performance of the algorithm, Monte Carlo simulation is performed by randomly generating \( a_i, s \). Forty runs are tested for each case. The statistics of performance measures are provided in Table 2.

Note that the objective function of Example 1 is not everywhere convex. The following example has a nonquadratic, strictly convex objective function, and its optimal solution can be specified by the user.

### Example 2.†

The system dynamics is the same as in Example 1. Stagewise Cost Function is given by

\[
g(x, u) = \exp(a_1 x) + \exp(b_1 u) - a_1 x \exp(a_1 x^*) - b_1 u \exp(b_1 u^*)
+ c_1(x - x^*)(x - x^*)
+ c_2(u - u^*)(u - u^*) + d_t, \tag{5.4}
\]

† This example and corresponding testing results are prepared by Mr Jian Shi.
The optimal control \( u^* \) is set to be \(-2\) when \( t \) is

test performed, with efficiency measure, this is not observed in Table

<table>
<thead>
<tr>
<th>( M \times T \times \xi_p )</th>
<th>( \xi_p )</th>
<th>( \xi_p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 3 \times 42 )</td>
<td>0.94</td>
<td>0.313</td>
</tr>
<tr>
<td>( 3 \times 84 )</td>
<td>1.12</td>
<td>0.373</td>
</tr>
<tr>
<td>( 3 \times 168 )</td>
<td>1.17</td>
<td>0.390</td>
</tr>
<tr>
<td>( 3 \times 336 )</td>
<td>1.25</td>
<td>0.417</td>
</tr>
<tr>
<td>( 7 \times 42 )</td>
<td>2.19</td>
<td>0.313</td>
</tr>
<tr>
<td>( 7 \times 84 )</td>
<td>2.19</td>
<td>0.313</td>
</tr>
<tr>
<td>( 7 \times 168 )</td>
<td>2.35</td>
<td>0.336</td>
</tr>
<tr>
<td>( 7 \times 336 )</td>
<td>2.15</td>
<td>0.307</td>
</tr>
</tbody>
</table>

where

\[
d_i = -\exp(a_i x_i^*) - \exp(b_i u_i^*)
+ a_i x_i^* \exp(a_i x_i^*) + b_i u_i^* \exp(b_i x_i^*).
\]

In (5.4), \( x_i^* \) and \( u_i^* \) are user-designed optimal solutions, \( a_i \) and \( b_i \) are coefficients, \( c_1, c_2 > 0 \) guarantee the strict convexity of \( J \), and dimensions of \( u_i \) and \( x_i \) are four.

Forty Monte Carlo simulation runs are performed, with \( \{a_i\} \) and \( \{b_i\} \) being two independent, identically distributed random sequences whose components are uniformly distributed over \([-0.03, 0.03]\). The values of \( c_1 \) and \( c_2 \) are both 0.0001. The initial state \( x_0 \) is 0.

The optimal control \( u_i^* \) is set to be \(-2\) when \( t \) is a multiple of 5, otherwise \( u_i^* = 0.5 \). The optimal states are then determined according to the system dynamics. The initial nominal controls are arbitrarily picked from the interval \([-11, 11]\).

Testing results are given in Table 3.

The above testing results for both examples show that the performance of the two-level algorithm improves as the number of stages increases. This indicates that the temporal decomposition has good potential in tackling long horizon problems under a parallel processing environment. However, we also observe that the efficiencies of the 7-processor cases are lower than those of the 3-processor cases. Though Table 2 shows a trend that the 7-processor case can outperform the 3-processor case in the efficiency measure, this is not observed in Table 3. Further investigation on the nature of the algorithm and more numerical testing are needed to draw specific conclusions on the relationship between efficiency and the number of processors.

6. CONCLUSIONS

In this paper, a new method is presented to solve long horizon optimal control problems. Its key features include decomposition along the time axis, coordination using initial and terminal states of subproblems, and parallel processing to take advantage of the availability of cost-effective parallel processing facilities. The two-level approach has nice properties in that the global optimum is not changed, and furthermore, the high level problem is a convex programming problem under appropriate conditions. A parallel, two-level optimization algorithm for unconstrained problems has been presented. It employs Newton's method at the high level and the Differential Dynamic Programming at the low level. The high level solution is open-loop in nature, whereas low level control is in linear variational feedback form. Numerical testing results show that our approach is feasible, the two-level algorithm is suitable for parallel processing, and its performances improve as the time horizon increases.

Note that although the NM-DDP algorithm is designed for unconstrained, convex problems, the two-level problem formulation and its properties (Sections 2 and 3) are addressed based on a larger class of problems. It is therefore believed that the ideas of time decomposition and coordination is promising in tackling very long horizon optimal control problems. A successful application of the approach depends on the availability of efficient two-level algorithms. The Newton-DDP combination presented here is an example for unconstrained, convex problems. The development of efficient algorithms for specific applications can be very problem dependent as for many other large-scale optimization techniques.

Acknowledgements—This work was supported in part by NSF Grants ECS-85-13613, ECS-85-04133, ECS-85-12815 and UCD faculty research grant 85-86. A preliminary version of the work was presented at the 1986 American Control Conference (Chang et al., 1986a). The authors would like to thank Mr Jian Shi in providing a test example for the paper.

REFERENCES

Let us also define the set of admissible controls for a given admissible state trajectory \((x_N, \ldots, x_i)\) as
\[
U_A = \{(u_{i-1}, \ldots, u_0) \mid (x_N, \ldots, x_i, u_{i-1}, \ldots, u_0) \in XU_A\}.
\] (A.4)

**Problem (P)'** is then equivalent to the following two-level problem (P-H)'

\[
\text{min} \, \text{min} \, J, \quad \text{subject to} \quad x_{n+1} = A_x x_n + B_x u_k, \quad k = (j - 1)T, \ldots, jT - 1,
\]

and \(x_{N-jT} = x_T\) given. (B.2)

In (P-H)', we choose all state variables as high level decision variables. If the set \(\{x_{jT}, j = 1, \ldots, M\}\) is chosen instead, we obtain (P-H) in (2.6) by following the same arguments. Note that in this case, the high level decision variables \(x_{jT}\) are to be chosen from the following feasible set:

\[
X_H = \{(x_{jT}, x_{(M-j)T}, \ldots, x_T) \mid \text{there exists a} \quad (x_N, x_{N-1}, \ldots, x_1) \in X_A\}. \tag{A.6}
\]

**APPENDIX B: DIFFERENTIAL DYNAMIC PROGRAMMING WITH FIXED TERMINAL STATES**

Consider a subproblem (P-1):

\[
J(x_{(j-1)T}, x_{jT}) = \min \sum_{k=1}^{jT-1} g_k(x_k, u_k), \tag{B.1}
\]

subject to \(x_{k+1} = A_k x_k + B_k u_k, \quad k = (j - 1)T, \ldots, jT - 1,\)

and \(x_{jT} = x_T\) given. (B.2)

\[
\begin{align*}
\delta x_{jT} &= A_{jT-1} \delta x_{jT-1} + B_{jT-1} \delta u_{jT-1}, \tag{B.3}
\end{align*}
\]

subject to \(\delta x_{jT} = A_{jT-1} \delta x_{jT-1} + B_{jT-1} \delta u_{jT-1}.\) (B.4)

The above approximate problem is a Quadratic Programming (QP) problem (Avriel, 1976) with linear equality constraint, where \(\delta x_{jT}\) and \(\delta u_{jT-1}\) are treated as given terms in the problem. The solution of the above QP problem is determined by the nature of the equality constraint (B.5). To convey main ideas, we assume that \(B_{jT-1}\) is invertible to simplify the handling of constraints caused by fixed terminal state. Analysis for the case where \(B_{jT-1}\) is not invertible can be found in Chang (1986). When \(B_{jT-1}\) exists, there is no need for QP. The unique solution is obtained by solving

\[
\delta u_{jT-1}^* = B_{jT-1}^{-1} \delta x_{jT-1} - A_{jT-1}^{-1} \delta x_{jT-1} = \alpha_{jT-1} + \beta_{jT-1} \delta x_{jT-1} + \gamma_{jT-1} \delta x_{jT-1}, \tag{B.6}
\]

where \(\alpha_{jT-1}, \beta_{jT-1}, \gamma_{jT-1}\) are called control coefficients.

Substituting this solution into (B.4), the cost-to-go function is then of the following form:

\[
\begin{align*}
\hat{V}_{jT-1}(\delta x_{jT-1}) &= \hat{V}_{jT-1}(\delta x_{jT-1}, \delta u_{jT-1}) = \hat{V}_{jT-1}(\delta x_{jT-1}, Q_{jT-1} \delta x_{jT} + \delta x_{jT} R_{jT-1} \delta x_{jT} + S_{jT-1} \delta x_{jT-1} + W_{jT-1} \delta x_{jT-1} + \delta x_{jT} + \theta_{jT-1}), \tag{B.7}
\end{align*}
\]

where \(P, Q, R, S, W,\) and \(\theta\) are appropriate coefficients.

For an intermediate stage \(k, (j - 1)T \leq k \leq jT - 2,\) by

**Time decomposition for optimal control**

85


applying quadratic approximation to \( g_k \), we have
\[
\bar{V}_k(\delta x_k, \delta x_T) = \min_{\delta u_k} \left[ \bar{g}_k(\delta x_k, \delta u_k) + \bar{V}_{k+1}(\delta x_{k+1}, \delta x_T) \right],
\] (B.8)
subject to \( \delta x_{k+1} = A_k \delta x_k + B_k \delta u_k \). (B.9)

By assuming that \( \bar{V}_k(\delta x_{k+1}, \delta x_T) \) is in a quadratic form similar to (B.7) and by substituting (B.9) into it, an unconstrained quadratic optimization problem is obtained:
\[
\bar{V}_k(\delta x_k, \delta x_T) = \min_{\delta u_k} \left[ \delta x_k^T C_k \delta x_k + \delta u_k^T D_k \delta x_k + \delta u_k^T E_k \delta u_k + F_k(\delta x_T) \delta x_k + G_k(\delta x_T) \delta u_k + \theta_k \right],
\] (B.10)
where \( F \) and \( G \) are affine functions of \( \delta x_T \). The optimal solution again has the following affine form:
\[
\delta u_k^* = \alpha_k + \beta_k \delta x_k + \gamma_k \delta x_T,
\] (B.11)
and \( \bar{V}_k \) turns out to be a quadratic function of \( \delta x_k \) and \( \delta x_T \).

Performing above steps backwards in time along the nominal trajectory, control coefficients \( \alpha_k, \beta_k \) and \( \gamma_k \) for all stages can be found. Since (B.10) is an approximation, a successor policy construction step is needed to guarantee that the actual cost associated with constructed controls is less than the original nominal cost. This step proceeds by letting
\[
u_k(\epsilon) = \bar{u}_k + \epsilon \alpha_k + \beta_k (x_k - \bar{x}_k),
\] (B.12)
and
\[
x_{k+1} = A_k x_k + B_k u_k(\epsilon), \quad k = (i-1)T, \ldots, jT - 1.
\] (B.13)
The initial value of \( \epsilon \) is one and the corresponding cost is evaluated. If the cost is lower than the nominal cost, the new policy is used to update the nominal trajectories. Otherwise, \( \epsilon \) is reduced by half till the constructed policy yields a cost lower than the nominal. The backward approximation and forward policy construction procedure are repeated until convergence is achieved. Since \( \bar{V}_k(T) \) is a quadratic approximation of the variation of \( J_k \) with respect to \( \delta x_T \), the gradient and Hessian of the low-level optimal cost with respect to \( \delta x_T \) and \( \delta x_T \) are thus readily available at convergence.