Automatica, Vol. 25, No. 1, pp. 77–86, 1989
Printed in Great Britain.

0005–1098/89 $3.00 + 0.00
Pergamon Press plc

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.

(Lasdon, 1970). The emphasis is on computational efficiency, and the nature of solutions is usually open-loop. On the other hand, for decentralized control applications, methods such as goal coordination, price coordination, interaction prediction, etc. (Findeisen *et al.*, 1980; Jamshidi, 1983; Shimizu and Ishizuka, 1985) have been developed to solve large-scale dynamic optimization problems along the line of decomposition and coordination. Their emphases are on the dynamic and information feedback aspects. In most of these approaches, decompositions are on state and/or control variables. 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 high level parameter optimization problem is created to determine the optimal coordination. In such a decomposition, subproblems are completely decoupled once coordination parameters are given, thus they can be solved in parallel.

The basic idea of the new method can be elaborated as follows. Consider a discrete-time, deterministic optimal control problem. If the optimal solution is known, its corresponding optimal state trajectory $\{x^\star(k)\}$ can be obtained. Let the optimal state trajectory be decomposed into M segments along the time axis, say, segment i starts from stage iT to stage $(i+1)T$, where T is a positive integer. The terminal state of segment i, $x^\star((i+1)T)$, is also the initial state of segment $i+1$. Define subproblem i as the optimization problem over the period $[iT, (i+1)T]$, having the corresponding part of system dynamics and cost function as the original problem, and $x^\star(iT)$ and $x^\star((i+1)T)$ as its initial and terminal states. From the optimality requirement, an optimal control and its corresponding state trajectory of the original problem in the period $[iT, (i+1)T]$ must be an optimal solution for subproblem i. Therefore, if the optimal states $\{x^\star(iT)\}_{i=1}^{M}$ are given, then subproblems can be solved in parallel since they are decoupled.

The problem then reduces to how to design an efficient algorithm to search for $\{x^\star(iT)\}_{i=1}^{M}$. To do this, a coordination center is formed at the high level. Its function is to supply subproblems with initial and terminal states. Once subproblems reach their individual solutions, they pass certain information back to the center. The center, based upon the information received, updates subproblems' initial and terminal states for the next iteration. The optimal solution can then be obtained iteratively under proper convexity conditions.

In this decomposition, the high level problem is a parameter optimization problem, as its goal is to find $\{x(iT)\}_{i=1}^{M}$ that minimizes the overall cost. Therefore, many computational methods for parameter optimization can be used to solve it. Similarly, many methods for optimal control can be used to solve low level subproblems. The selection of both methods should be carefully made based on the desired nature of the solution and the computational efficiency of the overall scheme. Note that as the high level is a parameter optimization problem, the solution of the two-level scheme cannot be completely closed-loop even if closed-loop solutions are achieved at the low level. Solutions with a nature between open-loop and closed-loop are generally expected. Note also that state decomposition can be carried out inside each subproblem when needed. In this sense, our approach is complementary to existing results.

To illustrate the ideas of our scheme, and to demonstrate the benefit of parallel processing for long horizon optimal control problems, a two-level optimization algorithm is developed for the time decomposition of a class of problems with convex, stagewise additive objective function and linear system dynamics. The algorithm consists of two basic components: the Differential Dynamic Programming (DDP) for low level subproblems and the Newton method for the high level (the NM-DDP algorithm). The DDP at the low level explicitly exploits system dynamics, results in variational feedback solutions, is efficient for standard optimal control problems (Jacobson and Mayne, 1970; Ohno, 1978; Yakowitz and Rutherford, 1984; Yakowitz, 1986), and generates first and second order derivative information needed for the high level optimization. The Newton method at the high level generates good search direction for coordination parameters, has quadratic convergence rate near the optimum, and has a special structure suitable for parallel processing as a consequence of time decomposition.

Lacking a parallel processor, our numerical testings were performed on an IBM mainframe with detailed timing and calculation of performmance measures (speedup and efficiency, to be discussed in Section 5). These performance measures are good for tightly coupled parallel processing systems where memory is shared among processors and interprocessor com-

munications 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:

$$\text{(P)} \min_{u_i} J, \text{ with } J = g_N(x_N) + \sum_{i=0}^{N-1} g_i(x_i, u_i), \tag{2.1}$$

subject to the system dynamics

$$x_{i+1} = f_i(x_i, u_i), \quad i = 0, \ldots, N-1, \text{ with } x_0 \text{ given}, \tag{2.2}$$

and constraints

$$x \in X \subset R_n, \quad \text{and} \quad u \in U \subset R^m. \tag{2.3}$$

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 $\{x_{jT}, j = 1, \ldots, M\}$, we can decompose (P) into M independent, T-stage subproblems as follows.

(P-j) *Subproblem* j, $j = 1, \ldots, M$

$$\min_{u_k} J_j, \quad \text{with} \quad J_j = \sum_{k=(j-1)T}^{jT-1} g_k(x_k, u_k), \tag{2.4}$$

subject to the system dynamics

$$x_{k+1} = f_k(x_k, u_k), \quad (j-1)T \le k \le jT-1, \quad \text{with } x_{(j-1)T} \text{ and } x_{jT} \text{ given}, \tag{2.5}$$

and the corresponding constraints of (2.3).

We assume for simplicity that (P-j) has an optimal solution if there exists a feasible sequence of controls that brings the system from $x_{(j-1)T}$ to x_{jT}. Denote $J_j^\star(x_{(j-1)T}, x_{jT})$ as its optimal cost. If $(x_{(j-1)T}, x_{jT})$ is not feasible, i.e. there does not exist a feasible sequence of controls to bring the system from $x_{(j-1)T}$ to x_{jT}, we let $J_j^\star = \infty$. The low level subproblem (P-j) is therefore well defined for all $(x_{(j-1)T}, x_{jT}) \in X^2$.

Since the solution of (P-j) depends upon $x_{(j-1)T}$ and x_{jT}, the set of variables $\{x_{jT}\}_{j=1}^M$ can be chosen as coordination parameters. A high level problem, called (P-H), is thus created to select the best coordination terms so that the total cost is minimized. That is,

$$\text{(P-H)} \min_{(x_{MT}, \ldots, x_T) \in X^M} J, \text{ with } J \equiv \sum_{j=1}^{M} J_j^\star(x_{(j-1)T}, x_{jT}) = g_N(x_N) + \sum_{i=0}^{N-1} g_i(x_i^\star, u_i^\star), \tag{2.6}$$

where $x_i^\star$ and $u_i^\star$ are solutions of (P-j) with $x_{(j-1)T}^\star \equiv x_{(j-1)T}$ and $x_{jT}^\star \equiv x_{jT}$, $j = 1, \ldots, M$.

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 $\{x_T\}_{j=1}^M$ 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 optima 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 $J_j^*(x_{(j-1)T}, x_{jT})$ is convex in terms of its arguments. We therefore consider subproblem (P-j). For simplicity of presentation, the subproblem index j is omitted and $x_{(j-1)T}$ is written as x_0 in the following proof. Define the set of admissible state and control trajectories XU_A as

$$XU_A \equiv \{(x_N, \ldots, x_1, u_{N-1}, \ldots, u_0) \in X^N \times U^N \mid (2.2) \text{ holds for } (x_N, \ldots, u_0)\}. \quad (3.1a)$$

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

$$X_{0T} \equiv \{(x_0, x_T) \mid \text{there exists a trajectory } (x_T, \ldots, x_1, u_{T-1}, \ldots, u_0) \in XU_A\}. \quad (3.1b)$$

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

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

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

$$x_{i+1} = A_i x_i + B_i u_i, \quad i = 0, \ldots, T-1, \quad (3.2)$$

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

$$x_T = \Phi_{-1} x_0 + \sum_{i=0}^{T-1} \Phi_i B_i u_i, \quad (3.3)$$

where

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

Let x_T^k, x_0^k and $(u_i^k, i = 0, \ldots, T-1)$, $k = 1, 2$, denote two sets of quantities satisfying (3.3). Define

$$x_0^0 \equiv \alpha x_0^1 + (1-\alpha) x_0^2, \quad (3.4a)$$

$$x_T^0 \equiv \alpha x_T^1 + (1-\alpha) x_T^2, \quad (3.4b)$$

$$u_i^0 \equiv \alpha u_i^1 + (1-\alpha) u_i^2, \quad i = 0, \ldots, T-1, \quad (3.4c)$$

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

$$\begin{aligned} x_T^0 &= \Phi_{-1}(\alpha x_0^1 + (1-\alpha) x_0^2) \\ &\quad + \sum_{i=0}^{T-1} \Phi_i B_i (\alpha u_i^1 + (1-\alpha) u_i^2) \\ &= \Phi_{-1} x_0^0 + \sum_{i=0}^{T-1} \Phi_i B_i u_i^0, \end{aligned} \quad (3.4d)$$

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

(b) For a given pair of (x_0^k, x_1^k), $k = 0, 1, 2$ as defined in part (a), denote $\{x_i^{*k}, u_i^{*k}\}$ 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

$$J^*(x_0^0, x_T^0) \le \alpha J^*(x_0^1, x_T^1) + (1-\alpha) J^*(x_0^2, x_T^2). \quad (3.5)$$

Define

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

Note that X and U are convex sets. By letting

$$\hat{x}_i = \alpha x_i^{*1} + (1-\alpha) x_i^{*2}, \quad \text{and} \quad \hat{u}_i = \alpha u_i^{*1} + (1-\alpha) u_i^{*2}, \quad (3.7)$$

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

$$\begin{aligned} J^*(x_0^0, x_T^0) &= \min_{(u_0, \ldots, u_{T-1}) \in U(x_0^0, x_T^0)} g_T(x_T) + \sum_{i=0}^{T-1} g_i(x_i, u_i) \\ &\le g_T(x_T) + \sum_{i=0}^{T-1} g_i(\hat{x}_i, \hat{u}_i) \\ &\le g_T(x_T) + \sum_{i=0}^{T-1} (\alpha g_i(x_i^{*1}, u_i^{*1}) \\ &\quad + (1-\alpha) g_i(x_i^{*2}, u_i^{*2})) \\ &= \alpha J^*(x_0^1, x_T^1) + (1-\alpha) J^*(x_0^2, x_T^2). \end{aligned} \quad (3.8)$$

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. $x_i \in X = R^n$ and $u_i \in U = R^n$. 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:

$$x_h^{k+1} = x_h^k - [\nabla^2 J(x_h^k)]^{-1} \nabla J(x_h^k), \qquad (4.1)$$

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

From (2.6), we have

$$\frac{\partial J}{\partial x_{jT}} = \frac{\partial J_j^*(x_{(j-1)T}, x_{jT})}{\partial x_{jT}} + \frac{\partial J_{j+1}^*(x_{jT}, x_{(j+1)T})}{\partial x_{jT}}. \qquad (4.2)$$

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:

$$\nabla^2 J = \begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & 0 & \\ & a_3 & b_3 & c_3 & & \\ & & \cdot & \cdot & b_{M-1} & \\ & 0 & & a_{M-1} & a_M & c_{M-1} \\ & & & & & b_M \end{bmatrix}, \qquad (4.3)$$

where $b_j \equiv \partial^2(J_j^* + J_{j+1}^*)/\partial x_{jT}^2$, $a_j \equiv \partial^2(J_{j-1}^* + J_j^*)/\partial x_{(j-1)T}\partial x_{jT}$ and $c_j \equiv \partial^2(J_{j+1}^* + J_{j+2}^*)/\partial x_{jT}\partial x_{(j+1)T}$. Note that both ∇J_j and $\nabla^2 J_j$ can be computed locally by the jth subproblem and passed onto the high level to form ∇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_{jT-1} is assumed to be invertible to simplify the handling of constraints caused by the fixed terminal state. Analysis for the case where B_{jT-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

$$y = [\nabla^2 J(x_h)]^{-1} \nabla J(x_h), \qquad (4.4)$$

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

$$[\nabla^2 J(x_h)] y = \nabla J(x_h). \qquad (4.5)$$

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

$$a_j y_{j-1} + b_j y_j + c_j y_{j+1} = v_j, \quad j = 1, \ldots, M, \quad a_1 = c_M = 0, \qquad (4.6)$$

where v_j is a vector consisting of the components of ∇J corresponding to the jth block. Assume for simplicity that $M = 2^{m+1} - 1$, 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 unknown y, 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

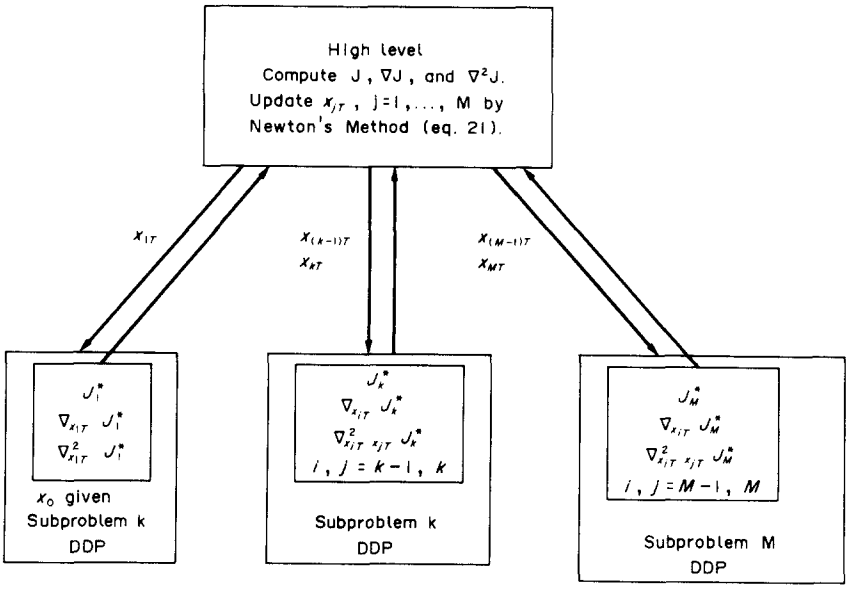

FIG. 1. The Newton-DDP algorithm with parallel structure.

reduction step. The one-block linear equation is then solved for y_{m+1}. The back-substitution step is performed in a reverse sequence of the reduction step. At each stage, the available result is substituted into the block tridiagonal equation constructed at the corresponding stage in the reduction step, and the unresolved variables in the equation can be computed easily and independently for different blocks. This backward procedure continues till the original problem is solved. The algorithm is suited for parallel computation, as many quantities involved can be computed independently. Note that the number of processors used is reduced by half after each reduction, and this may not be desirable for the consideration of efficiency. There are other parallel algorithms for solving block tridiagonal equations (Kowalik and Kumar, 1985), and the selection of a proper algorithm depends on the architecture of the parallel processor to be used.

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:

$$J = \sum_{t=1}^{MT} \left[\sum_{i=1}^{n} (x_{it} - a_{it})^2 \sum_{i=1}^{m} u_{it}^2 + \sum_{i=1}^{m} u_{it}^2 + \sum_{i=1}^{m} \sum_{j>i}^{m} u_{it} u_{jt} + 100 \sum_{i=1}^{n} (x_{it} - a_{it})^2 \right], \quad (5.1)$$

where $m = n = 4$, and $\{a_{it}\}$ is an independent, identically distributed random sequence with uniform distribution over the interval $[-2, 2]$. This cost function is designed to let x_{it} close to a_{it} and have small controls. It is convex when magnitudes of x and u are not too large.

System dynamics:

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

$$\text{and } B = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}. \quad (5.2)$$

TABLE 1. TEST RESULTS FOR TWO PERFORMANCE MEASURES

	DDP		Newton-DDP				
Stages	T_1	f.c.	$M \times T$	I_p	S_p	E_p	f.c.
42	0.59	0.1764597	3 × 14	0.36	1.639	0.546	0.1764654
84	1.12	0.3430279	3 × 28	0.61	1.836	0.612	0.3430364
168	2.21	0.7067726	3 × 56	1.12	1.973	0.658	0.7068281
336	4.41	1.240006	3 × 112	2.15	2.051	0.684	1.240177
42			7 × 6	0.31	1.90	0.271	0.1764609
84			7 × 12	0.43	2.605	0.372	0.3430302
168			7 × 24	0.73	3.027	0.432	0.7068199
336			7 × 48	1.16	3.80	0.543	1.240194

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

The initial state is $x_0 = [0\,0\,0\,0]'$, and the initial nominal control is $\bar{u}_{it} = 0$ for all i and t.

Convergence criteria

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

$$|J_j^{k+1} - J_j^k|/J_j^k \leq \eta. \tag{5.3}$$

The convergence threshold η is 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 kth 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_ks. 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 $S_p \equiv T_1/(T_p + T_h)$, measures improvement in computation time by using parallel processing. On the other hand, efficiency $E_p \equiv S_p/M$, measures how well the processing power is utilized. Testing results for a sequence of $\{a_{it}\}$ are given in Table 1.

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

M	$M \times T$	$\bar{S}_p$	σ_s^2	$\bar{E}_p$	σ_e^2 (10^{-2})	$\bar{A}_c$ (10^{-5})	σ_c^2 (10^{-10})
3	42	1.35	0.293	0.448	3.25	0.325	0.044
3	84	1.37	0.049	0.455	0.551	0.463	0.037
3	168	1.54	0.074	0.513	0.819	0.638	0.189
3	336	1.56	0.052	0.519	0.579	0.005	0.193
7	42	2.21	0.942	0.315	1.92	0.854	0.109
7	84	2.70	0.143	0.385	0.292	1.32	0.431
7	168	3.12	0.311	0.446	0.635	0.010	0.041
7	336	3.26	0.118	0.465	0.242	0.178	0.052

$\bar{S}_p$: mean of S_p, σ^2: variance.
$\bar{A}_c$: 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_{it}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

$$\begin{aligned} g_t(x_t, u_t) = {} & \exp(a_t'x_t) + \exp(b_t'u_t) \\ & - a_t'x_t \exp(a_t'x_t^\star) - b_t'u_t \exp(b_t'u_t^\star) \\ & + c_1(x_t - x_t^\star)'(x_t - x_t^\star) \\ & + c_2(u_t - u_t^\star)'(u_t - u_t^\star) + d_t, \end{aligned} \tag{5.4}$$

† This example and corresponding testing results are prepared by Mr Jian Shi.

TABLE 3. STATISTICS FOR THE PERFORMANCE OF EXAMPLE 2 BASED ON MONTE CARLO SIMULATION

M	$M \times T$	$\bar{S}_p$	$\bar{E}_p$
3	42	0.94	0.313
3	84	1.12	0.373
3	168	1.17	0.390
3	336	1.25	0.417
7	42	2.19	0.313
7	84	2.19	0.313
7	168	2.35	0.336
7	336	2.15	0.307

where

$$d_t = -\exp(a_t' x_t^\star) - \exp(b_t' u^\star) + a_t' x_t^\star \exp(a_t' x_t^\star) + b_t' u_t^\star \exp(b_t' x_t^\star).$$

In (5.4), $x_t^\star$ and $u_t^\star$ are user-designed optimal solutions, a_t and b_t are coefficients, $c_1, c_2 > 0$ guarantee the strict convexity of J, and dimensions of u_t and x_t are four.

Forty Monte Carlo simulation runs are performed, with $\{a_t\}$ and $\{b_t\}$ 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 $u_t^\star$ is set to be -2 when t is a multiple of 5, otherwise $u_t^\star = 0.5$. 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 could 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 testings 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 is then 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 many 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

Avriel, M. (1976). *Nonlinear Programming Analysis and Methods*. Prentice-Hall, Englewood Cliffs, NJ.

Chang, T. S. and P. B. Luh (1985). On the decomposition and coordination of large scale dynamic control problems. *Proc. 24th Conf. on Decision and Control*, Fort Lauderdale, Florida, pp. 1484–1485, December 1985.

Chang, S. C. (1986). A hierarchical, temporal decomposition approach to long horizon optimal control problems. Ph.D. Thesis, Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT, December 1986.

Chang, S. C., P. B. Luh and T. S. Chang (1986a). A parallel algorithm for temporal decomposition of long horizon optimal control problems. *Proc. 1986 Control and Decision Conf.*, Athens, Greece, December 1986.

Chang, T. S., S. C. Chang and P. B. Luh (1986). A hierarchical decomposition for large scale optimal control problems with parallel processing capability. *Proc. 1986*

American Control Conf., Seattle, Washington, June 1986, pp. 1995–2000.
Findeisen, W., F. N. Bailey, M. Bryds, K. Malinowski, P. Tatjewski and A. Wozniak (1980). *Control and Coordination in Hierarchical Systems*. John Wiley, New York.
Geoffrion, A. M. (1972). Generalized benders' decomposition. *J. Opt. Applic.*, **10**, 237–260.
Heller, D. (1976). Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems. *SIAM J. Numer. Anal.*, **13**, 484–496.
Heller, D. (1978). A survey of parallel algorithms in numerical linear algebra. *SIAM Rev.*, **20**, 740–777.
Jacobson, D. and D. Mayne (1970). *Differential Dynamic Programming*. Elsevier, New York.
Jamshidi, M. (1983). *Large-scale Systems*. North-Holland, New York.
Kowalik, J. S. and S. P. Kumar (1985). Parallel algorithms for recurrence and tridiagonal equations. In J. S. Kowalik (Ed.), *Parallel MIMD Computation: HEP Supercomputer and Its Applications*. MIT Press, Cambridge, MA.
Lasdon, L. S. (1970). *Optimization Theory for Large Systems*. Macmillan, London.
Ohno, K. (1978). A new approach to differential dynamic programming for discrete time systems. *IEEE Trans. Aut. Control*, **AC-23**, 37–47.
Polak, E. (1973). A historical survey of computational methods in optimal control. *SIAM Rev.*, **15**, 553–583.
Sage, A. P. and C. C. White, III (1977). *Optimum Systems Control* (2nd ed.). Prentice-Hall, Englewood Cliffs, NJ.
Scheel, C. (1981). Parallel processing of optimal control problems by dynamic programming. *Inf. Sci.*, **25**, 85–114.
Shapiro, J. F. (1979). *Mathematical Programming, Structures and Algorithms*. John Wiley, New York.
Shimizu, K. and Y. Ishizuka (1985). Optimality conditions and algorithms for parameter design problem with two-level structure. *IEEE Trans. Aut. Control*, **AC-30**, 986–993.
Silverman, G. (1972). Primal decomposition of mathematical programs by resource allocation, *Ops Res.*, **20**, 58–93.
Yakowitz, S. (1986). The stagewise Kuhn–Tucker condition and differential dynamic programming. *IEEE Trans. Aut. Control*, **AC-31**, 25–30.
Yakowitz, S. and B. Rutherford (1984). Computational aspects of discrete-time optimal control. *Appl. Math. Comput.*, **15**, 29–45.

APPENDIX A: PROOF OF THEOREM 1

To show that problems (P) and (P-H) have the same global optimum, we shall first examine the original problem (P). In (P), an admissible control sequence $\{u_i,\ i=0,\ldots,N-1\}$ generates an admissible state trajectory. The optimization can therefore be thought of as being carried out over all possible pairs of admissible controls and corresponding state trajectories. Let us define the set of admissible state and control trajectories XU_A as

$$XU_A \equiv \{(x_N,\ldots,x_1,u_{N-1},\ldots,u_0)\in X^N\times U^N \mid (2.2) \text{ holds for } (x_N,\ldots,u_0)\}, \quad (A.1)$$

then problem (P) is equivalent to the parameter optimization problem (P)′ defined below

$$(P)'\quad \min_{(x_N,\ldots,u_0)\in XU_A} J, \quad \text{with} \quad J=g_N(x_N)+\sum_{i=0}^{N-1} g_i(x_i,u_i). \quad (A.2)$$

It is well known that a parameter minimization problem can be converted into a two-level minimization problem by first minimizing over a subset of the parameters, and then minimizing over the remaining parameters. To apply to problem (P)′, let us first define the set of all admissible state trajectories X_A as

$$X_A \equiv \{(x_N,\ldots,x_1) \mid \text{there exists a } (x_N,\ldots,x_A,u_{N-1},\ldots,u_0)\in XU_A\}. \quad (A.3)$$

Let us also define the set of admissible controls for a given admissible state trajectory $(x_N,\ldots,x_1)$ as

$$U_{A|X}(x_N,\ldots,x_1) \equiv \{(u_{N-1},\ldots,u_0) \mid (x_N,\ldots,x_1,u_{N-1},\ldots,u_0)\in XU_A\}. \quad (A.4)$$

Problem (P)′ is then equivalent to the following two-level problem (P-H)′:

$$(P\text{-}H)'\quad \min_{(x_N,\ldots,x_1)\in X_A}\ \min_{(u_{N-1},\ldots,u_0)\in U_{A|X}} J, \quad \text{with} \quad J\equiv g_N(x_N)+\sum_{i=0}^{N-1} g_i(x_i,u_i). \quad (A.5)$$

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:

$$X_H \equiv \{(x_{MT},x_{(M-1)T},\ldots,x_T) \mid \text{there exists a } (x_N,x_{N-1},\ldots,x_1)\in X_A\}. \quad (A.6)$$

APPENDIX B: DIFFERENTIAL DYNAMIC PROGRAMMING WITH FIXED TERMINAL STATES

Consider a subproblem (P-j):

$$J_j(x_{(j-1)T},x_{jT}) \equiv \min_{u_k} \sum_{k=(j-1)T}^{jT-1} g_k(x_k,u_k) \quad (B.1)$$

$$\text{subject to}\quad x_{k+1}=A_kx_k+B_ku_k,\ k=(j-1)T,\ldots,jT-1, \quad \text{and } x_{(j-1)T} \text{ and } x_{jT} \text{ given.} \quad (B.2)$$

Let $\{\bar{u}_k,\ k=(j-1)T,\ldots,jT-1\}$ be a given set of nominal controls, and $\{\bar{x}_k,\ k=(j-1)T,\ldots,jT\}$ be the corresponding state trajectory. By taking the Taylor series expansion of the above objective function at the last stage, we formulate an approximate linear quadratic problem:

$$\hat{V}_{jT-1}(\delta x_{jT-1},\delta x_{jT}) \equiv \min_{\delta u_{jT-1}} \hat{g}_{jT-1}(\delta x_{jT-1},\delta u_{jT-1}), \quad (B.3)$$

with

$$\hat{g}\equiv \delta x'_{jT-1}C_{jT-1}\delta x_{jT-1}+\delta u'_{jT-1}D_{jT-1}\delta x_{jT-1} +\delta u'_{jT-1}E_{jT-1}\delta u_{jT-1}+F_{jT-1}\delta x_{jT-1}+G_{jT-1}\delta u_{jT-1}. \quad (B.4)$$

$$\text{subject to}\quad \delta x_{jT}=A_{jT-1}\delta x_{jT-1}+B_{jT-1}\delta u_{jT-1}. \quad (B.5)$$

The above approximate problem is a Quadratic Programming (QP) problem (Avriel, 1976) with linear equality constraint, where δx_{jT} and δx_{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}^{-1} exists, there is no need for QP. The unique solution is obtained by solving (B.5):

$$\delta u^{\star}_{jT-1}=B^{-1}_{jT-1}\delta x_{jT}-B^{-1}_{jT-1}A_{jT-1}\delta x_{jT-1} \equiv \alpha_{jT-1}+\beta_{jT-1}\delta x_{jT-1}+\gamma_{jT-1}\delta x_{jT-1}, \quad (B.6)$$

where α_{jT-1}, β_{jT-1} and γ_{jT-1} are called control coefficients. Substituting this solution into (B.4), the cost-to-go function is then of the following form:

$$\hat{V}_{jT-1}(\delta x_{jT-1},\delta x_{jT-1})=\delta x'_{jT-1}P_{jT-1}\delta x_{jT-1}+\delta x'_{jT-1}Q_{jT-1}\delta x_{jT} +\delta x'_{jT}R_{jT-1}\delta x_{jT}+S_{jT-1}\delta x_{jT-1}+W_{jT-1}\delta x_{jT}+\theta_{jT-1}, \quad (B.7)$$

where P, Q, R, S, W and θ are appropriate coefficients.

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

applying quadratic approximation to g_k, we have

$$\hat{V}_k(\delta x_k, \delta x_{jT}) \equiv \min_{\delta u_k} [\hat{g}_k(\delta x_k, \delta u_k) + \hat{V}_{k+1}(\delta x_{k+1}, \delta x_{jT})], \tag{B.8}$$

$$\text{subject to } \delta x_{k+1} = A_k \delta x_k + B_k \delta u_k. \tag{B.9}$$

By assuming that $\hat{V}_{k+1}(\delta x_{k+1}, \delta x_{jT})$ is in a quadratic form similar to (B.7) and by substituting (B.9) into it, an unconstrained quadratic optimization problem is obtained:

$$\hat{V}_k(\delta x_k, \delta x_{jT}) = \min_{\delta u_k} [\delta x_k' C_k \delta x_k + \delta u_k' D_k \delta x_k + \delta u_k' E_k \delta u_k + F_k(\delta x_{jT}) \delta x_k + G_k(\delta x_{jT}) \delta u_k + \theta_{k+1}], \tag{B.10}$$

where F and G are affine functions of δx_T. The optimal solution again has the following affine form:

$$\delta u_k^{\star} = \alpha_k + \beta_k \delta x_k + \gamma_k \delta x_{jT}, \tag{B.11}$$

and $\hat{V}_k$ turns out to be a quadratic function of δx_k and δx_{jT}.

Performing above steps backwards in time along the nominal trajectory, control coefficients α_k, β_k and γ_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

$$u_k(\epsilon) = \bar{u}_k + \epsilon \alpha_k + \beta_k (x_k - \bar{x}_k), \tag{B.12}$$

and

$$x_{k+1} = A_k x_k + B_k u_k(\epsilon), \quad k = (j-1)T, \ldots, jT - 1. \tag{B.13}$$

The initial value of ϵ 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, ϵ 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 $\hat{V}_{(j-1)T}(\delta x_{(j-1)T}, \delta x_{jT})$ is a quadratic approximation of the variation of $J_j^{\star}$ with respect to $\delta x_{(j-1)T}$ and δx_{jT}, the gradient and Hessian of the low-level optimal cost with respect to $\delta x_{(j-1)T}$ and δx_{jT} are thus readily available at convergence.