INT. J. CONTROL, 1992, VOL. 56, NO. 6, 1275–1297

A new parallel algorithm for optimal control problems of interconnected systems

XIAOHONG GUAN† and PETER B. LUH†

Parallel algorithms play a crucial role in utilizing parallel computers to overcome the difficulties of computation intensive problems. This paper presents a new parallel algorithm for solving optimal control of discrete-time interconnected systems. The idea is to use Lagrange multipliers to relax the coupling among subsystems. The decomposed subproblems are solved in parallel by using the differential dynamic programming (DDP) method. The Lagrange multipliers are selected as coordinating variables and updated at the high level to ensure global optimality. A method for estimating the computational complexity of the decomposition/coordination approach is presented, and the relationship between speed-up and major system parameters is established. It is shown that significant speed-ups are difficult to achieve by using conventional optimization techniques at high level. A large number of high level iterations may offset the reduction in computation for dealing with smaller decomposed subproblems in parallel. The parallel variable metric (PVM) method is found to be a promising high level algorithm for loosely coupled systems. Numerical results show that comparing with one level DDP, the PVM/DDP algorithm obtains significant speed-ups under a simulated parallel processing environment. Moreover global variational feedback controls which are invaluable for on-line control systems are obtained.

1. Introduction

Computational requirements have long been one of the major concerns for the optimal control of large-scale interconnected systems. The advent of parallel computers fostered a considerable amount of effort to solve these problems by using parallel algorithms to speed up computation. One approach is based on the idea of decomposition and coordination, where a large optimal control problem is decomposed into a number of subproblems, and appropriate coordinating variables are introduced. This forms a two-level structure, where the low level consists of many smaller optimal control subproblems, and the high level is a parameter optimization problem. With the coordinating variables given, low level subproblems are decoupled and solved in parallel. The global optimality is then ensured by iteratively updating coordinating variables at the high level.

Successful methods for solving long-horizon problems have been developed along the line of time decomposition, where the original problem is decomposed along the time axis (Chang *et al.* 1989, Tang *et al* 1991). Several spatial decomposition techniques, including mixed coordination and goal coordination, have been developed for interconnected systems by relaxing coupling among subsystems using Lagrange multipliers (Pearson 1971, Singh 1980, Jamshidi 1983, Perry 1984). In mixed coordination, coupling variables and Lagrange multipliers are selected as coordinating variables; whereas in goal coordination

Received 10 August 1991. Revised 30 January 1992.

† Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06269-3157, U.S.A.

only Lagrange multipliers are selected. The major difficulties of spatial decomposition, in contrast to time decomposition, lie in the fact that the dimension of high-level coordinating variables is generally proportional to the product of the dimension of coupling variables and time horizon. Since the number of high-level iterations usually increases with the above product, the reduction in computational requirements for solving lower dimensional subproblems in parallel may be offset by a large number of high-level iterations. Little work has been found in the literature on analysing whether decomposition/coordination methods are better than undecomposed ones in terms of computational complexity and efficiency. Moreover, existing techniques generally do not provide global feedback controls.

This paper presents a new parallel algorithm for solving large scale, discrete-time optimal control of interconnected systems by using the spatial decomposition and goal coordination scheme. The problem formulation and the decomposition/coordination framework are presented in § 2. In § 3, the differential dynamic programming (DDP) method of Yakowits and Rutherford (1984) for solving low-level subproblems and a few potential techniques for high level parameter optimization are discussed. A method for estimating the computational complexity of the decomposition and coordination approach is presented, and a relationship between algorithm speed-up and major system parameters is established. Our analysis shows that significant speed-ups are difficult to achieve by using conventional parameter optimization techniques at the high level.

To increase speed-up, parallel optimization techniques have to be adopted at the high level. Our complexity analysis indicates that the parallel variable metric (PVM) method reported in Straeter (1973), Van Laarhoven (1985) and Loostma and Ragsdell (1988) is promising among several candidates for loosely coupled systems. The PVM/DDP algorithm developed in § 4 is efficient with a quadratic convergence rate near optimal at both levels. Moreover, global variational feedback controls which are invaluable for on-line control systems are obtained. The PVM/DDP algorithm is implemented in a simulated parallel processing environment. Numerical testing results presented in § 5 indicate that, comparing with one level DDP, significant speed-ups are obtained.

2. Problem formulation

Consider the following dynamic system consisting of N interconnected subsystems:

$$x(t+1)\, f_i(x(t), u(t)), \quad t = 0, \ldots, T-1 \tag{2.1}$$

where

$$x(t) = [x_1^{\mathrm{T}}(t), x_2^{\mathrm{T}}(t), \ldots, x_N^{\mathrm{T}}(t)]^{\mathrm{T}}$$

$$u(t) = [u_1^{\mathrm{T}}(t), u_2^{\mathrm{T}}(t), \ldots, u_N^{\mathrm{T}}(t)]^{\mathrm{T}}$$

The variable $x_i(t) \in \mathbb{R}^{n_i}$ is the state of the ith subsystem with $\sum_{i=1}^{N} n_i = n$, $u_i(t) \in \mathbb{R}^{m_i}$ is the control of the ith subsystem with $\sum_{i=1}^{N} m_i = m$, and superscript T denotes transpose. The system dynamics are assumed to have the following structure:

$$f_i(x(t), u(t)) = \begin{bmatrix} f_{1t}(x_1(t), u_1(t)) + C_1(t)z_1(t) \\ \cdots \\ f_{it}(x_i(t), u_i(t)) + C_i(t)z_i(t) \\ \cdots \\ f_{Nt}(x_N(t), u_N(t)) + C_N(t)z_N(t) \end{bmatrix} \tag{2.2}$$

In (2.2), the dynamics of each subsystem consist of a linear or nonlinear function $f_{it}(\cdot)$ of its own state and control, and a linear coupling term $C_i(t)z_i(t)$ from other subsystems. The function $f_{it}(\cdot)$ is assumed to have continuous second-order derivatives with respect to $x_i(t)$ and $u_i(t)$. The coupling of the ith subsystem from other subsystems is denoted by $z_i(\cdot)$ and is linear in state variables, i.e.

$$z_i(t) \equiv \sum_{\substack{j=1 \\ j\neq i}}^{N} M_{ij}(t)x_j(t), \quad t = 0, \ldots, T-1, \quad i = 1, \ldots, N \tag{2.3}$$

with $z_i(t) \in \mathbb{R}^{r_i}$ and $\sum_{i=1}^{N} r_i = r$. In the above, matices M_{ij} and C_i are of appropriate dimensions. For a loosely coupled system, these matrices can be properly defined so that $r_i < n_i$, and this leads to $r \ll n$.

The cost function to be minimized is of the following additive form:

$$J \equiv \sum_{i=1}^{N} \left[\sum_{t=0}^{T-1} g_{it}(x_i(t), u_i(t), z_i(t)) + g_{iT}(x_i(T)) \right] \tag{2.4}$$

where $g_{it}(\cdot)$ and $g_{iT}(\cdot)$ are assumed to have continuous second-order derivatives with respect to their arguments. Furthermore, $g_{it}(\cdot)$ is assumed to be convex, and have positive definite Hessian with respect to $u_i(\cdot)$ and $z_i(\cdot)$. The problem is then (**P**):

$$\min_{u(t)} J \tag{2.5}$$

subject to (2.1), with the initial condition $x(0)$ given.

The spatial decomposition idea is to partition **P** into N smaller subproblems and solve them in parallel. To do so, coupling among subsystems is relaxed by using the Lagrange multiplier sequence

$$\lambda(t) = [\lambda_1^{\mathrm{T}}(t), \lambda_2^{\mathrm{T}}(t), \ldots, \lambda_N^{\mathrm{T}}(t)]^{\mathrm{T}}, \quad t = 0, 1, \ldots, T-1 \tag{2.5}$$

where

$$\lambda_i(t) \in R^{r_i}, \quad i = 1, \ldots, N, \quad \text{and} \quad \lambda(t) \in R^r$$

Then the lagrangian can be written as

$$L = \sum_{i=1}^{N} \left[\sum_{t=0}^{T-1} \left[g_{it}(x_i(t), u_i(t), z_i(t)) + \lambda_i^{\mathrm{T}}(t)(z_i(t) - \sum_{\substack{j=1 \\ j\neq i}}^{N} M_{ij}(t)x_j(t)) \right] + g_{iT}(x_i(T)) \right] \tag{2.6}$$

Rearrange the lagrangian according to subsystem index, it can be rewritten as

$$L = \sum_{i=1}^{N} \left[\sum_{t=0}^{T-1} \left[g_{it}(x_i(t), u_i(t), z_i(t)) + \lambda_i^{\mathrm{T}}(t)z_i(t) - \sum_{\substack{j=1 \\ j\neq i}}^{N} \lambda_j^{\mathrm{T}} M_{ji}(t)x_i(t) \right] + g_{iT}(x_i(T)) \right] \tag{2.7}$$

For notational convenience, define a stack form of the Lagrange multipliers with dimension $r \cdot T$ as:

$$\lambda_s = [\lambda^T(0), \lambda^T(1), \ldots, \lambda^T(T-1)]^T \tag{2.8}$$

By using the duality theorem (Luenberger 1984) and selecting Lagrange multipliers as coordinating variables, low level optimal control subproblems are formulated as follows:

$(P-i)$, $i = 1, 2, \ldots, N$:

$$\min_{\substack{u_i(t)\\ z_i(t)}} L_i(\lambda_s), \text{ with } L_i(\lambda_s) \equiv \sum_{t=0}^{T-1}\Big[g_{it}(x_i(t), u_i(t), z_i(t)) + \lambda_i^T(t)z_i(t) - \sum_{\substack{j=1\\ j\neq i}}^{N}\lambda_j^T(t)M_{ji}(t)x_i(t)\Big] + g_{iT}(x_i(T)) \tag{2.9}$$

subject to

$$x_i(t+1) = f_{it}(x_i(t), u_i(t)) + C_i(t)z_i(t) \tag{2.10}$$

Let $x_i^*(t)$, $u_i^*(t)$ and $z_i^*(t)$ denote, respectively, the optimal state, control and coupling variables of $P-i$ for a given λ_s, and $L_i^*(\lambda_s)$ the corresponding lagrangian. Then the high level problem $P-H$ is to select λ_s so as to maximize the dual function:

$(P-H)$:

$$\max_{\lambda_s} \Phi(\lambda_s), \quad \text{with} \quad \Phi(\lambda_s) \equiv \sum_{i=1}^{n} L_i^*(\lambda_s) \tag{2.11}$$

Note that $P-H$ is a parameter optimization problem where the dimension of λ_s is $r \cdot T$.

3. Low-level and high-level algorithms and complexity analysis

3.1. *The low-level differential dynamic programming algorithm*

Many methods are available for solving low level subproblems, especially for the linear quadratic (LQ) case (Bryson and Ho 1975, Singh 1980, Jamshidi 1983). However, it is difficult to find an efficient algorithm for problems with general object functions and system dynamics. Furthermore, most numerical methods do not provide feedback controls which are required for most on-line control systems.

In this paper, the DDP method is used to solve low-level subproblems. DDP is a dynamic-programming based successive quadratic approximation method. Suppose that the initial nominal state, control and coupling variables, $\{\bar{x}_i^0(t), \bar{u}_i^0(t), \bar{z}_i^0(t), t = 0, 1, \ldots, T-1\}$, are given. The method consists of a backward sweep determining variational feedback control coefficients, and a forward sweep updating the nominal trajectory. The DDP algorithm of Yakowits and Rutherford (1984) is modified to include coupling and coordinating variables in the backward and forward sweep process. Coupling variables $\{z_i(\cdot)\}$ are treated as control variables to the ith relaxed subsystem.

The backward sweep is based on the quadratic approximations of the stage-wise costs, the optimal cost-to-go functions, and system dynamics. By minimizing the sum of the stage-wise and optimal cost-to-go at each stage, linear variational feedback controls can be obtained:

$$\delta u_i(t) = \alpha_{1i}(t) + \beta_{1i}(t)\delta x_i(t) \tag{3.1}$$

$$\delta z_i(t) = \alpha_{2i}(t) + \beta_{2i}(t)\delta x_i(t), \quad t = 0, 1, \ldots, T-1 \tag{3.2}$$

This backward sweep starts from the terminal stage and works backwards in time, until the initial stage is reached. The detailed derivations are provided in Appendix A.

In the forward sweep, variational feedback control coefficients are used to update nominal variables as follows:

$$\bar{u}_i^{l+1}(t) = \bar{u}_i^l(t) + v\alpha_{1i}(t) + \beta_{1i}(t)(\bar{x}_i^{l+1}(t) - \bar{x}_i^l(t)) \tag{3.3}$$

$$\bar{z}_i^{l+1}(t) = \bar{z}_i^l(t) + v\alpha_{2i}(t) + \beta_{2i}(t)(\bar{x}_i^{l+1}(t) - \bar{x}_i^l(t)) \tag{3.4}$$

$$\bar{x}_i^{l+1}(t+1) = f_i(\bar{x}_i^{l+1}(t), \bar{u}_i^{l+1}(t)) + C_i(t)\bar{z}_i^{l+1}(t) \tag{3.5}$$

where $\bar{x}_i^l(t)$ is the nominal state at the lth iteration with $\bar{x}_i^l(0) = x_i(0)$ given, etc. This forward sweep starts with the initial stage and works forward in time, until the terminal stage is reached. The value of v is first set to one, and reduced by half if necessary until the cost of the new trajectory is lower than that of the original one. This v reduction step is similar to the line search procedure in parameter optimization.

The backward sweep and forward sweep iterate until the cost function in (2.9) cannot be improved. It has been shown that the DDP algorithm has quadratic convergence rate near optimal (Yakowits and Rutherford 1984). The final nominal trajectory is optimal for the given coordinating variables $\{\lambda(t), t = 0, 1, \ldots, T-1\}$ if the system dynamics are linear and the cost function is convex. Otherwise, the result is locally optimal.

An important feature of the DDP algorithm is that the variational feedback controls obtained in the backward sweep can be used in on-line control operations. When nominal optimal controls are applied to the system in the absence of disturbance, the state variables will move along the nominal optimal trajectory. If, for some reasons, the state trajectory deviates from the nominal one, the variational controls can be applied to pull it back. Consider the variational optimal control subproblem which is a linearized version of subproblem $P-i$ with $x_i(t)$, $u_i(t)$ and $z_i(t)$ replaced by $\delta x_i(t)$, $\delta u_i(t)$ and $\delta z_i(t)$. It is shown in the following theorem that for an LQ problem, variational feedback controls obtained in (3.1) and (3.2) are optimal for the variational optimal control subproblem.

Theorem 3.1: *Suppose that the system dynamics* (2.2) *are linear, and the stage-wise cost functions* (2.4) *are of the following quadratic form*:

$$g_{it}(x_i(t), u_i(t), z_i(t)) = \tfrac{1}{2}[x_i^{\mathrm{T}}(t)Q_i(t)x_i(t) + u_i^{\mathrm{T}}(t)R_i(t)u_i(t) + z_i^{\mathrm{T}}(t)S_i(t)z_i(t)] \tag{3.6}$$

$$g_{iT}(x_i(T)) = \tfrac{1}{2}x_i^{\mathrm{T}}(T)Q_i(T)x_i(T) \tag{3.7}$$

where $Q_i(\cdot)$ *are positive semidefinite,* $R_i(\cdot)$ *and* $S_i(\cdot)$ *are positive definite. Then*

the variational feedback controls $\delta u_i(t)$ and $\delta z_i(t)$ obtained in (3.1) *and* (3.2) *are optimal for the variational control subproblem, and the close loop system formed by applying these controls is stable.*

The proof of Theorem 3.1 is given in Appendix B. If the stage-wise cost functions are not quadratic and/or the system dynamics are not linear, (3.1) and (3.2) can be regarded as the first-order approximations to the optimal variational controls.

The variational variable $\delta z_i(t)$ in (3.2) is actually not an independent control to subsystem i during on-line operations. Rather, it is a linear combination of variational states of other subsystems. To obtain global variational feedback, $\delta u_i(t)$ should be a function of both $\delta x_i(t)$ and $\delta z_i(t)$. Based on the first order necessary conditions for optimal variational controls, it can be shown that $\delta u_i(t)$ can be expressed as follows:

$$\delta u_i(t) = \alpha_i(t) + \beta_i(t)\delta x_i(t) + \gamma_i(t)\delta z_i(t) \tag{3.8}$$

Derivations are given in Appendix A. Thus, global variational feedback controls required for most on-line control systems are obtaind.

All the computations involved in one iteration of DDP are matrix and vector manipulations. The complexity of matrix–matrix multiplication and matrix inversion is proportional to the cube of the matrix size for square matrices (Press *et al.* 1988), and is the dominating factor. The complexity of matrix–vector manipulation is proportional to the product of the dimensions of the matrix, and is negligible in this case. Thus, the complexity of the ith subproblem is

$$C_{\text{ddpi}} = s_{\text{ddp}} N_{\text{ddpi}} T n_i^3 \tag{3.9}$$

where s_{ddp} is a scaling constant reflecting the types and numbers of matrix operations, N_{ddpi} is the product of the total number of iterations and the average number of v reduction steps per iteration for the ith subproblem. Suppose that low-level subproblems are solved in parallel with N processors. Then the complexity of the low level is

$$C_{\text{ddp}} = \max_i C_{\text{ddpi}} = s_{\text{ddp}} N_{\text{ddp}} T n_{\text{ddp}}^3 \tag{3.10}$$

where N_{ddp} is the N_{ddpi} of the worst subproblem, and n_{ddp} is the corresponding state dimension.

If DDP is used to solve problem P directly without decomposition, the complexity of one level DDP is

$$C_0 = s_0 N_0 T n^3 \tag{3.11}$$

where s_0 is a scaling constant as in (3.9), N_0 is the product of the total number of iterations and average number of v reduction steps per iteration. From the backward sweep of DDP in Appendix A, it should be clear that the one-level DDP algorithm cannot make good use of the sparsity of the system dynamics. Even for an LQ problem, the sparsity of the system matrices generally does not lead to the sparsity of hessian matrices of cost-to-go functions in the backward sweep. This can be seen from the computational procedure of the backward sweep of DDP.

The number of DDP iterations is generally problem dependent. For LQ

problems, however, DDP can be completed in just one iteration. This is because DDP has the property of Newton's method. By comparing (3.10) and (3.11), potential savings in computation may be obtained in dealing with subproblems of smaller dimensions. The major difficulties of most spatial decomposition and coordination methods, however, lie in the fact that the number of high-level iterations generally increases with the dimension of high-level coordinating variables. The reduction in complexity for dealing with smaller subproblems may be offset by a large number of high-level iterations. Selection of a high-level algorithm is thus of crucial importance to obtain significant speed-ups and is discussed next.

3.2. *Typical high level algorithms*

Newton's method is efficient for solving parameter optimization problems, however, it requires the inverse of the required hessian matrix. This is difficult for $P-H$ because $\{x_i^*(t), u_i^*(t), z_i^*(t)\}$ obtained from $P-i$ are not explicit functions of λ_s. Furthermore, inverting a big hessian matrix of $\Phi(\lambda_s)$ with dimension $rT \times rT$ may result in numerical instability. Our concentration is therefore on the conjugate gradient and variable metric methods which are more attractive among others.

The conjugate gradient method updates λ_s as follows (Luenberger 1984):

$$\lambda_s^{k+1} = \lambda_s^k + \eta s^k \tag{3.12}$$

where

$$s^k = -\nabla\Phi(\lambda_s^k) + \frac{\|\Phi(\lambda_s^k)\|^2}{\|\nabla\Phi(\lambda_s^{k-1})\|^2} s^{k-1} \tag{3.13}$$

is the conjugate gradient direction vector, k is the iteration index and η is the step size. From (2.10), $\nabla\Phi(\lambda_s)$ can be easily obtained once low level subproblems are solved:

$$\nabla\Phi(\lambda_i(t)) = z_i^*(t) - \sum_{\substack{i=1 \\ i\neq j}}^{N} M_{ij}(t)x_j^*(t) \tag{3.14}$$

Thus, the complexity of one high-level iteration using the conjugate gradient algorithm is $O((rT)^2)$ resulting from matrix–vector manipulations. The complexity of the combined conjugate gradient and DDP algorithm is therefore given by

$$C_{cg} = N_{cg}[s_{cg}(rT)^2 + C_{ddp}] \tag{3.15}$$

where s_{cg} is a scaling constant reflecting the number of matrix–vector manipulations, and N_{cg} is the product of the total number of high-level iterations and the average number of function evaluations in a high-level line search step.

The basic idea of the variable metric method is to approximate the inverse hessian by using variables and gradients between two consecutive iterations (Luenberger 1984). This eliminates the major difficulty of Newton's method for requiring the inversion of high-dimensional hessian matrices. Suppose that the approximate inverse hessian at the kth iteration is given by H^k, then the equation for updating H^k is

$$H^{k+1} = H^k + \tau^k(\rho^k)(\rho^k)^T \tag{3.16}$$

where

$$\rho^k = \delta_k - H^k y^k \tag{3.17}$$

$$\delta^k = \lambda_s^{k+1} - \lambda_s^k \tag{3.18}$$

$$y^k = \nabla\Phi(\lambda_s^{k+1}) - \nabla\Phi(\lambda_s^k) \tag{3.19}$$

$$\tau^k = ((\rho^k)^T(y^k))^{-1} \tag{3.20}$$

with $H^0 = I$, the identity matrix. Then, following Newton's method, coordinating variables are updated according to

$$\lambda_s^{k+1} = \lambda_s^k - \eta H^k \nabla\Phi(\lambda_s^k) \tag{3.21}$$

where η is the step size. Similar to the analysis of the conjugate gradient method, the complexity of one high-level iteration using the variable metric method is of order $O((rT)^2)$. The total complexity is

$$C_{vm} = N_{vm}[s_{vm}(rT)^2 + C_{ddp}] \tag{3.22}$$

where s_{vm} is a scaling constant, N_{vm} is the product of total number of high-level iterations and average number of function evaluations in a high-level line search step.

The number of high-level iterations is generally problem dependent. To compare the performance of different algorithms, some assumptions have to be made. It will be shown next that the number of high level iterations can be obtained under the LQ assumption, and formulas for speed-ups can be explicitly derived. First, it is shown in the following theorem that high level object functions are quadratic for LQ problems.

Theorem 3.2: *Suppose that the system dynamics* (2.2) *are linear, and stage-wise cost functions* (2.4) *are of the quadratic forms as in* (3.6) *and* (3.7), *then the high level dual function* $\Phi(\lambda_s)$ *of* (2.11) *is of the following quadratic form*:

$$\Phi(\lambda_s) = \lambda_s^T \Gamma \lambda_s + \zeta^T \lambda_s + \text{constant} \tag{3.23}$$

with Γ *being negative definite*.

Proof: The proof is given in Appendix C. □

It is known that for the quadratic function (3.23), the number of iterations by using either the conjugate gradient or the variable metric method equals the dimension of λ_s, i.e. rT (Luenberger 1984). There is no formula for the average number of function evaluations per line search. From experience, it is assumed that the average number of function evaluations per line search is four. The scaling constants are determined based on the fact that the complexity of multiplying two square matices is one times the cube of the matrix size, matrix inversion is 4/3 times the cube of matrix size (Press *et al.* 1988, p. 38). Then, by counting the number of domianting matrix manipulations and neglecting non-dominating ones in each algorithm, the following scaling constants are obtained:

$$s_0 \approx 10, \quad s_{ddp} \approx 50, \quad s_{cg} \approx 1, \quad s_{vm} \approx 2 \tag{3.24}$$

With the above parameters we have:

$$C_0 = 20Tn^3 \tag{3.25}$$

$$C_{cg} = 4rT[(rT)^2 + 100Tn_{ddp}^3] \tag{3.26}$$

$$C_{vm} = 4rT[2(rT)^2 + 100Tn_{ddp}^3] \tag{3.27}$$

Note that speed-ups for two-level algorithms are obtainable if n_{ddp}, r, and T are small. That is, the system is finely decomposed (n_{ddp} is much smaller than n), loosely coupled (r small), and has a short time horizon. Even for such a system, speed-up may not be significant. Suppose $n_{ddp} = \frac{1}{15}n$, $r = \frac{1}{15}n$, then speed-ups are given by

$$S_{cg} = \frac{C_0}{C_{cg}} = \frac{15^4}{T[3T + 20n]} \tag{3.28}$$

$$S_{vm} = \frac{C_0}{C_{vm}} = \frac{15^4}{T[6T + 20n]} \tag{3.29}$$

Speed-ups are achievable only for small n and T, which is inconsistent with our original goal for solving problems with large dimensions. For example, with $n = 150$, $N = 10$, $n_{ddp} = 10$, $r = 10$ and $T = 10$, the speed-ups are $S_{cg} = 1{\cdot}67$, and $S_{vm} = 1{\cdot}65$, not significant for a ten processor system.

The above example clearly shows that the reduction in complexity by solving lower-dimensional subproblems in parallel is offset by large numbers of high-level iterations and function evaluations for the high level algorithms considered. Thus, reducing the number of high-level iterations and function evaluations is crucial for obtaining significant speed-ups.

4. The high level parallel variable metric algorithm

The key to improving the two-level algorithm is to reduce the number of high-level iterations and the number of function evaluations per iterations. This is possible by parallelizing the high-level algorithm.

One version of parallel conjugate gradient methods concurrently evaluates several function values in a line search (Loostma and Ragsdell 1988). The improvement, however, is not significant since the number of high-level iterations cannot be reduced. The parallel variable metric (PVM) method of Straeter (1973), Laarhoven (1985) and Loostma and Ragsdell (1988) is more attractive because it reduces the number of iterations and also does not require many independent line searches. Under the LQ assumption, the algorithm has the property of Newton's method, i.e. it converges in just one iteration (two if solving low-level subproblems for the given initial condition is included) and no line search is needed.

Let λ_s^k be the high level coordinating variable at the kth iteration, and H^k the corresponding approximate inverse hessian. The key idea of the PVM algorithm is to form rT linearly independent vectors around λ_s^k and to compute the gradients of Φ at these points in parallel. The approximate inverse hessian is then updated by using these gradients. Let

$$\delta^j = \varepsilon \cdot e^j \quad j = 1, \ldots, rT \tag{4.1}$$

where ε is a sufficiently small real number and e^j is the jth unit vector whose elements are zeros except for the jth element being one. Let

$$\lambda_s^{kj} \equiv \lambda_s^k + \delta^j, \quad j = 1, \ldots, rT \tag{4.2}$$

then λ_s^{kj}, $j = 1, \ldots, rT$, are rT linearly independent vectors around λ_s. Define the partial updating matrix as follows:

$$\Psi^{kj} \equiv \Psi^{k(j-1)} + \tau^{kj}(\rho^{kj})(\rho^{kj})^{\mathrm{T}}, \quad j = 1, \ldots, rT \tag{4.3}$$

where

$$\rho^{kj} = \delta^j - \Psi^{k(j-1)} y^{kj}, \quad j = 1, \ldots, rT \tag{4.4}$$

$$y^{kj} = \nabla\Phi(\lambda_s^{kj}) - \nabla\Phi(\lambda_s^k), \quad j = 1, \ldots, rT \tag{4.5}$$

$$\tau^{kj} = ((\rho^{kj})^{\mathrm{T}}(y^{kj}))^{-1}, \quad j = 1, \ldots, rT \tag{4.6}$$

The initial value of Ψ^{kj} is $\Psi^{k0} = H^k$. Then the approximate inverse hessian is obtained as a result of updating Ψ^{kj}:

$$H^{k+1} = \Psi^{k(rT)} \tag{4.7}$$

Finally, λ_s^k is updated according to (3.21). As mentioned in Loostma and Ragsdell (1988), the PVM algorithm has the property of quadratic convergence.

Given $\nabla\Phi(\lambda_s^{kj})$, the computations involved in PVM are rT sets of matrix–vector manipulations of dimension rT. This has the complexity of $O((rT)^3)$. Calculating $\nabla\Phi(\lambda_s^{kj})$ by using (3.14), however, requires the optimal state and coupling variables of all low-level subproblems. For each λ_s^{kj}, $j = 1, \ldots, rT$, N processors are needed to solve N low-level subproblems. Therefore, NrT processors are required for the parallel computation of $\nabla\Phi(\lambda_s^{kj})$.

To reduce the number of processors needed, low-level subproblems have to be grouped. The idea is to use one processor to solve problem $P - i$ for the given λ_s^k and λ_s^{kj}, $j = 1, \ldots, rT$, without increasing the computation by rT times. To do this, problem $P - i$ is first solved for λ_s^k. The solutions of $P - i$ corresponding to λ_s^{kj}, $j = 1, \ldots, rT$, can then be obtained as a by-product in the process, and the additional computations needed are not great. The reason is that for LQ problems the hessians of stage-wise cost functions are constant matrices. Therefore, quadratic terms of cost-to-go and optimal cost-to-go functions at each stage need only be calculated once for all the $(rT + 1)\lambda_s$. For non-LQ problems, hessians obtained for λ_s^k can also be used for λ_s^{kj}, $j = 1, \ldots, rT$, assuming ε is sufficiently small. Testing results presented in § 5 justify this approximation. This leads to solving N subproblems by using N processors in parallel, while each subproblem is solved by one processor for the given λ_s and λ_s^{kj}, $j = 1, \ldots, rT$. If more than N processors are available, further parallelization can also be done.

Now consider the complexity of the PVM/DDP algorithm. The additional computations at the low level beyond (3.10) are rT sets of matrix–vector manipulations in the backward and forward sweeps with complexity $O(rTn_{\mathrm{ddp}}^2)$. The complexity of the PVM/DDP algorithm is therefore

$$C_{\mathrm{pvm}} = N_{\mathrm{pvm}}[s_{\mathrm{vh}}(rT)^3 + N_{\mathrm{ddp}}T(s_{\mathrm{ddp}}n_{\mathrm{ddp}}^3 + s_{\mathrm{vl}}rTn_{\mathrm{ddp}}^2)] \tag{4.8}$$

where s_{vh} and s_{vl} are scaling constants, and N_{pvm} is the product of the total number of iterations and average number of function evaluations in a high-level line search step. Under the LQ assumption, $N_{\mathrm{pvm}} = 2$. Selecting scaling constants by counting the number of operations as $s_{\mathrm{vh}} \approx 8$, $s_{\mathrm{vl}} \approx 200$, one obtains

$$C_{\text{pvm}} = 2[8(rT)^3 + 2T(50n_{\text{ddp}}^3 + 200rTn_{\text{ddp}}^2)] \tag{4.9}$$

Then the speed-up for the PVM/DDP algorithm is given by

$$S_{\text{pvm}} = \frac{C_0}{C_{\text{pvm}}} = \frac{5Tn^3}{4(rT)^3 + T(50n_{\text{ddp}}^3 + 200rTn_{\text{ddp}}^2)} \tag{4.10}$$

Suppose $n_{\text{ddp}} = \frac{1}{15}n$, $r = \frac{1}{15}n$ as in § 3.2, (4.10) becomes

$$S_{\text{pvm}} = \frac{5 \times 15^3}{4T^2 + 200T + 50} \tag{4.11}$$

Note that S_{pvm} is independent of n, but decreases as T increases. For the example of § 3.2, one obtains $S_{\text{pvm}} = 6{\cdot}9$ for ten processors. The speed-up is significant.

From (4.10) and (4.11), it should be clear that the algorithm is suitable for finely decomposed, loosely coupled systems with short time horizons. As mentioned in § 3.1, one level DDP can hardly make good use of these properties to reduce the computational complexity. If time horizon T is too large, the combination of the time decomposition methods developed in Chang *et al.* (1989), Tang *et al.* (1991) and the spatial decomposition method presented here seems promising.

5. Implementation issues and simulation results

5.1. *Implementation issues and the simulated parallel processing environment*

Major issues for implementing a parallel algorithm on a parallel processing system include partitioning of computational tasks, allocation of the tasks to processors, and communication and synchronization among tasks residing in different processors. Task partitioning for lagrangian relaxation algorithms is generally obvious and simple. For the PVM/DDP algorithm, parallel computational tasks are the resolution of low-level subproblems. Suppose that there are N processors for solving N low-level subproblems in parallel one for each subproblem, and one of the N processors is also used for updating multipliers at high level.

At the beginning of each high-level iteration, the rT multiplier vectors λ_s^{kj} are down-loaded to every low-level task. After the low level subproblems are solved, rT sets of state and coupling variables are up-loaded to the high-level task and the multipliers are updated. There is no need for communication among low-level tasks. Inter-task synchronization is also simple and straightforward. Updating multipliers at high-level should not be started until all the subproblems have been solved, and all the local state and coupling variables are up-loaded. The updated multipliers can be broadcast to low-level tasks which solve subproblems in parallel without the need for synchronization.

Although the PVM/DDP algorithm can be implemented on any parallel processing system with no restriction on its architecture, the communication and synchronization overheads may vary for different environments. Based on the requirements analysed above, suitable architectures can be identified. Since there is no communication between low-level tasks, a tightly-coupled message-

passing system is more suitable than a shared memory system as the latter may result in conflicting access to the common multipliers by low-level tasks. Furthermore, a star architecture with a broadcasting mechanism for the central node or a bus architecture facilitates down-loading the multipliers. Therefore, tightly-coupled message-passing parallel processing systems with star or bus connecting topology are promising for the PVM/DDP algorithm.

One way to consider the magnitude of the communication overhead in implementing the PVM/DDP algorithm on the above architectures is to examine the ratio of the communication time to the computation time. It is reasonable to assume that the communication time is proportional to the size of the data vector transferred (Bertsekas and Tsitsiklis 1989). The time required to down-load rT multiplier vectors each with size rT is, therefore, $O((rT)^2)$. A low-level task may up-load its local state and coupling variables while other low-level tasks are still executing. However, low-level tasks may also request communication at the same time. Assume, for simplicity, that the local state and coupling variables of subsystems are up-loaded sequentially under a deterministic communication mechanism. There are rT sets of state and coupling variables, and the communication time need is $O((n+r)T \cdot rT)$. Assume further that communication operations are slower by an order of magnitude than computation operations for a tightly coupled message-passing system. Based on the above analysis and (4.9) under the LQ assumption, the ratio of communication time to computation time is then given by

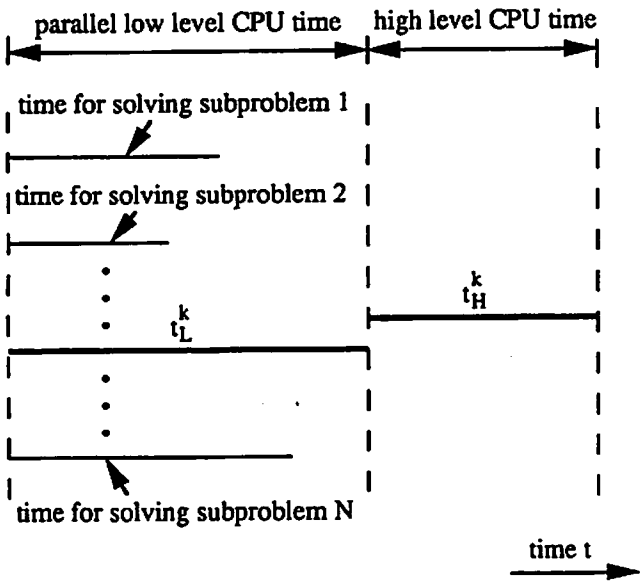
$$\frac{T_{\text{COMM}}}{T_{\text{COMP}}} = \frac{\text{communication time per iteration}}{\text{computation time per iteration}} = \frac{10[(rT)^2 + (n+r)rT^2]}{8(rT)^3 + 2T(50n_{\text{ddp}}^3 + 200rTn_{\text{ddp}}^2)}$$

For the examples to be presented with $n = 31$, $r = 2$, $n_{\text{ddp}} = 3$ and T ranging from 3 to 20, the ratio $T_{\text{COMM}}/T_{\text{COMP}}$ is less than 10%. Communication time for each high-level iteration is therefore not a dominating factor for the example to be considered.

The testing is performed on an IBM 3090 mainframe computer under a simulated parallel processing environment with a lack of a parallel processing system. The number of processors is assumed to be the same as the number of subsystems. Communication time is ignored based on the above analysis. The low-level CPU time at iteration k is obtained as the longest CPU time t_{L}^k in solving individual subproblems for that iteration, as shown in the Figure. The total CPU time is then taken as the sum of the high-level time plus low-level time for all iterations, i.e. $\sum_{k=1}^{K}(t_{\text{L}}^k + t_{\text{H}}^k)$, where t_{H}^k is high-level CPU time in iteration k and K is the total number of high level iterations.

5.2. *Simulated numerical results*

Testing of decomposition/coordination algorithms for large scale optimal control problems is not an easy job. There is no standard for system structure, degree of coupling, etc upon which testing of algorithms can be based. Furthermore, dimensions of test problems should be large enough so that advantages of parallel decomposition and coordination algorithms can stand out. This, however, is costly. In fact, few examples have been found in the literature



Simulated parallel CPU time at high level iteration k.

that compare performance of decomposed algorithms versus undecomposed ones.

In this section, one problem with a quadratic cost function and one problem with a non-quadratic cost function are tested. The high-level convergence criterion is

$$\nabla\Phi(\lambda_s) \leqslant 0{\cdot}0001 \tag{5.1}$$

The convergence criterion for low-level DDP is

$$\frac{|L_i^{l+1} - L_i^l|}{|L_i^l| + 1} \leqslant 0{\cdot}00001 \tag{5.2}$$

with $L_i \equiv J$ for one level DDP.

The first problem tested is an LQ problem. The system dynamics and cost function are given in Appendix D. The major system parameters are: $n = 31$, $m = 11$, $N = 11$, $r = 2$ and $n_{ddp} = 3$. Testing results are summarized in Table 1.

	One-level DDP			PVM/DDP				
T	IT_1†	CPU time (s)	Cost function	IT_h‡	CPU time (s)	Cost function	Speed-up	S_{pvm}§
3	1	0·36	184·12	2	0·035	184·12	10·3	12·1
6	1	1·23	276·48	2	0·18	276·48	6·83	6·17
10	2	2·16	298·11	2	0·66	298·11	3·27	3·67
15	2	3·42	304·37	2	1·82	304·38	1·84	2·38
20	2	4·57	306·34	2	3·74	306·35	1·22	1·86

† Number of one level DDP iterations.
‡ Number of high-level iterations.
§ Estimated speed-ups from (4.10).

Table 1. Testing results for Problem 1 with a quadratic cost function.

It can be seen from Table 1 that significant speed-ups are obtained by using the PVM/DDP algorithm for problems with short time horizons. Actual speed-ups are close to estimated ones obtained according to (4.10). Note that for $T = 10$, 15 and 20, the number of one-level DDP iterations is two instead of one, the theoretical value for LQ problems. This may be caused by accumulated numerical errors since the sizes of matrices in one-level DDP are large in these cases. The degree of coupling also affects the speed-up. For example, if the dimensions of state and control variables are reduced but the dimensions of coupling variables are kept the same, a similar but more tightly coupled system is formed with $n = 20$, $m = 10$, $N = 10$ and $r = 2$. The resulting speed-up for $T = 6$ is 1·6 versus 6·83 in Table 1.

The second problem is identical to the first except that quartic terms with significant weightings are added to the quadratic cost function. This cost function is given in Appendix D. Results are summarized in Table 2.

The speed-ups in Table 2 are better than those in Table 1. This may be caused by the fact that quadratic approximations in DDP for small subproblems are more accurate than those for large undecomposed problems. Thus, the one level DDP may require more iterations to converge. For example, with $T = 6$, the number of DDP iterations is 23 for the undecomposed problem; while only 4 or 5 for each decomposed subproblem. A large number of one-level DDP iterations also indicates that weightings of non-quadratic terms are significant. Furthermore, the efficiency for $T = 3$ is $15{\cdot}1/11 = 1{\cdot}37$, which is greater than one (efficiency is defined as the ratio of speed-up versus the number of processors, and is a measure of how computation power is utilized). This says that even by using one processor, the PVM/DDP algorithm in this case is better than the one-level DDP in terms of computation time.

Another factor affecting speed-ups is the value of ε in (4.1). If ε is too small, the differences between high level variables λ_s^{kj} and λ_s^k and their corresponding gradients may be too small. This may result in numerical difficulties since the approximate inverse hessian may not be accurate. On the other hand, ε cannot be too large. As mentioned in § 4, hessians of stage-wise cost and cost-to-go functions corresponding to λ_s^k are used to approximate those corresponding to λ_s^{kj}. Large ε could cause inaccuracy of these approximations. Testing for different values of ε is performed on Problem 2 with $T = 6$. The results are summerized in Table 3.

Testing results validate our reasoning above. The value $\varepsilon = 0{\cdot}002$ is used in generating Table 1 and Table 2.

	One-level DDP			PVM/DDP			
T	IT_1†	CPU time (s)	Cost function	IT_h‡	CPU time (s)	Cost function	Speed-up
3	11	1·96	182·64	3	0·13	182·64	15·1
6	23	9·38	270·21	3	0·87	270·21	10·7
10	37	26·50	300·59	5	6·30	300·59	4·20
15	63	59·86	307·98	4	16·52	307·98	3·62

† Number of one level DDP iterations.
‡ Number of high-level iterations.

Table 2. Results for Problem 2 with a non-quadratic cost function.

ε	IT_h	Cost function	CPU time (s)	Speed-up
0·001	4	270·2087	1·14	8·23
0·002	3	270·2087	0·87	10·7
0·003	6	270·2092	1·78	5·27

Table 3. Results for different values of ε.

6. Summary

A new parallel algorithm for solving optimal control of large-scale interconnected systems based on decomposition and coordination is developed. In the process, potential algorithms are analysed and compared in terms of computational complexity. This sheds a new light on the performance of parallel optimal control algorithms. It is found that reductions of high-level iterations and function evaluations are crucial for obtaining significant speed-ups. The PVM/DDP algorithm developed obtains signficant speed-ups, and is suitable for finely decomposed, loosely coupled systems with short time horizons. Furthermore, by utilizing the features of DDP, global variational feedback controls, valuable for on-line control systems, are obtained.

ACKNOWLEDGMENT

The work was supported in part by the National Science Foundation under Grant ECS-8717167. The authors would like to thank Professor S. C. Chang of the National Taiwan University for valuable comments.

Appendix A

Derivation of the backward sweep on the DDP algorithm

The DDP algorithm of Yakowits and Rutherford (1984) is modified to incorporate coupling and coordinating variables and to obtain global variational feedback controls. Consider the linear and quadratic parts of the Taylor series expansion of $g_{it}(x_i(t),\ u_i(t),\ z_i(t)),\ t = 0, 1, \ldots,\ T-1$, in (2.4) along a given nominal trajectory,

$$\begin{aligned} QP[g_{it}(x_i(t),\ u_i(t),\ z_i(t))] = {} & \delta x_i^{\mathrm{T}}Q_i(t)\delta x_i(t) + \delta u_i^{\mathrm{T}}(t)R_i(t)\delta u_i(t) \\ & + \delta z_i^{\mathrm{T}}(t)S_i(t)\delta z_i(t) + \delta u_i^{\mathrm{T}}(t)P_i(t)\delta x_i(t) \\ & + \delta z_i^{\mathrm{T}}(t)W_i(t)\delta x_i(t) + \delta u_i^{\mathrm{T}}(t)Y_i(t)\delta z_i(t) \\ & + q_i^{\mathrm{T}}(t)\delta x_i(t) + r_i^{\mathrm{T}}(t)\delta u_i(t) \\ & + s_i^{\mathrm{T}}(t)\delta z_i(t) \\ & + \text{constant} \end{aligned} \tag{A 1}$$

and

$$QP[g_{iT}(x_i(T))] = \delta x_i^{\mathrm{T}}(T)E_i(t)\delta x_i(t) + \mu_i(T)^{\mathrm{T}}\delta x_i(T) \tag{A 2}$$

where QP denotes the quadratic approximation operation. Assume that $g_{it}(x_i(t), u_i(t), z_i(t))$ is convex over $x_i(t)$, $u_i(t)$ and $z_i(t)$, $g_{iT}(x_i(T))$ is convex over $x_i(T)$, and $R_i(t)$ and $S_i(t)$ are positive definite. For simplicity, the system dynamics are assumed to be linear as follows:

$$f_i(x_i(t), u_i(t)) = A_i(t)x_i(t) + B_i(t)u_i(t) \tag{A 3}$$

Then the dynamics for the ith subsystem becomes:

$$x_i(t+1) = A_i(t)x_i(t) + B_i(t)u_i(t) + C_i(t)z_i(t) \tag{A 4}$$

The results can be easily extended to nonlinear systems following the derivation of Yakowits and Rutherford (1984).

In the backward sweep of DDP, one starts with the variational cost-to-go at state $T-1$.

$$\begin{aligned}
&V_i(T-1)\\
&= QP[g_{iT}(x_i(T)) + g_{i(T-1)}(x_i(T-1), u_i(T-1), z_i(T-1))\\
&\quad + \lambda_i^T(T-1)z_i(T-1) - \sum_{\substack{j=1\\ j\neq i}}^{N}\lambda_j^T(T-1)M_{ji}(T-1)x_i(T-1)]\\
&= \delta x_i^T(T-1)D_{1i}(T-1)\delta x_i(T-1) + \delta u_i^T(T-1)D_{2i}(T-1)\delta u_i(T-1)\\
&\quad + \delta z_i^T(T-1)D_{3i}(T-1)\delta z_i(T-1) + \delta u_i^T(T-1)D_{4i}(T-1)\delta x_i(T-1)\\
&\quad + \delta z_i^T(T-1)D_{5i}(T-1)\delta x_i(T-1) + \delta u_i^T(T-1)D_{6i}(T-1)\delta z_i(T-1)\\
&\quad + d_{1i}^T(T-1)\delta x_i(T-1) + d_{2i}^T(T-1)\delta u_i(T-1) + d_{3i}^T(T-1)\delta z_i(T-1)\\
&\quad + \text{constant}
\end{aligned} \tag{A 5}$$

where

$$D_{1i}(T-1) = Q_i(T-1) + A_i^T(T-1)E_i(T)A_i(T-1) \tag{A 6}$$

$$D_{2i}(T-1) = R_i(T-1) + B_i^T(T-1)E_i(T)B_i(T-1) \tag{A 7}$$

$$D_{3i}(T-1) = S_i(T-1) + C_i^T(T-1)E_i(T)C_i(T-1) \tag{A 8}$$

$$D_{4i}(T-1) = P_i(T-1) + B_i^T(T-1)[E_i(T) + E_i^T(T)]A_i(T-1) \tag{A 9}$$

$$D_{5i}(T-1) = W_i(T-1) + C_i^T(T-1)[E_i(T) + E_i^T(T)]A_i(T-1) \tag{A 10}$$

$$D_{6i}(T-1) = Y_i(T-1) + B_i^T(T-1)[E_i(T) + E_i^T(T)]B_i(T-1) \tag{A 11}$$

$$d_{1i}^T(T-1) = q_i^T(T-1) - \sum_{\substack{j=1\\ j\neq i}}^{N}\lambda_j^T(T-1)M_{ji}(T-1) + \mu_i^T(T)A_i(T-1) \tag{A 12}$$

$$d_{2i}^T(T-1) = r_i^T(T-1) + \mu_i^T(T)B_i(T-1) \tag{A 13}$$

$$d_{3i}^T(T-1) = s_i^T(T-1) + \lambda_i^T(T-1) + \mu_i^T(T)C_i(T-1) \tag{A 14}$$

The first-order necessary conditions for $\delta\mu_i(T-1)$ and $\delta z_i(T-1)$ are:

$$\begin{aligned}
\frac{\partial V_i(T-1)}{\partial \delta u_i(T-1)} &= 2D_{2i}(T-1)\delta u_i(T-1) + D_{6i}(T-1)\delta z_i(T-1) + d_{2i}(T-1)\\
&\quad + D_{4i}(T-1)\delta x_i(T-1) = 0
\end{aligned} \tag{A 15}$$

$$\begin{aligned}
\frac{\partial V_i(T-1)}{\partial \delta z_i(T-1)} &= 2D_{3i}(T-1)\delta z_i(T-1) + D_{6i}^T(T-1)\delta u_i(T-1)\\
&\quad + d_{3i}(T-1) + D_{5i}(T-1)\delta x_i(T-1) = 0
\end{aligned} \tag{A 16}$$

From (A 15) and (A 16), linear variational feedback controls are obtained:

$$\delta u_i(T-1) = \alpha_{1i}(T-1) + \beta_{1i}(T-1)\delta x_i(T-1) \tag{A 17}$$

$$\delta z_i(T-1) = \alpha_{2i}(T-1) + \beta_{1i}(T-1)\delta x_i(T-1) \tag{A 18}$$

where coefficients in the equations are appropriately defined. By substituting (A 16) and (A 17) into (A 15), the optimal cost-to-go at stage $T-1$ is given by

$$\bar{V}_i(T-1) = \delta x_i^{\mathrm{T}}(T-1)E_i(T-1)\delta x_i(T-1) + \mu_i^{\mathrm{T}}(T-1)\delta x_i(T-1) + \text{constant}, \tag{A 19}$$

where

$$\begin{aligned} E_i(T-1) &= \beta_{1i}^{\mathrm{T}}(T-1)D_{2i}(T-1)\beta_{1i}(T-1) + \beta_{2i}^{\mathrm{T}}(T-1)D_{3i}(T-1)\beta_{2i}(T-1) \\ &\quad + \beta_{1i}^{\mathrm{T}}(T-1)D_{6i}(T-1)\beta_{2i}(T-1) + \beta_{2i}^{\mathrm{T}}(T-1)D_{5i}(T-1)D_{1i}(T-1) \\ &\quad + \beta_{1i}^{\mathrm{T}}(T-1)D_{4i}(T-1) \end{aligned} \tag{A 20}$$

$$\begin{aligned} \mu_i^{\mathrm{T}}(T-1) &= d_{1i}^{\mathrm{T}}(T-1) + \alpha_{1i}^{\mathrm{T}}(T-1)[D_{2i}(T-1) + D_{2i}^{\mathrm{T}}(T-1)]\beta_{1i}(T-1) \\ &\quad + \alpha_{2i}^{\mathrm{T}}(T-1)[D_{3i}(T-1) + D_{3i}^{\mathrm{T}}(T-1)]\beta_{2i}(T-1) \\ &\quad + \alpha_{1i}^{\mathrm{T}}(T-1)D_{4i}(T-1) + \alpha_{2i}^{\mathrm{T}}(T-1)D_{5i}(T-1) \\ &\quad + \alpha_{1i}^{\mathrm{T}}(T-1)D_{6i}(T-1)\beta_{2i}(T-1) + \alpha_{2i}^{\mathrm{T}}(T-1)D_{6i}^{\mathrm{T}}(T-1)\beta_{2i}(T-1) \\ &\quad + d_{2i}^{\mathrm{T}}(T-1)\beta_{1i}(T-1) + d_{3i}^{\mathrm{T}}(T-1)\beta_{2i}(T-1) \end{aligned} \tag{A 21}$$

The cost-to-go at stage t, $t = 0, 1, \ldots, T-2$, is

$$\begin{aligned} V_i(t) = QP[&g_{ii}(x_i(t), u_i(t), z_i(t)) + \lambda_i^{\mathrm{T}}(t)z_i(t) \\ &- \sum_{\substack{j=1 \\ j\neq i}}^{N} \lambda_j^{\mathrm{T}}(t)M_{ji}(t)x_i(t) + \bar{V}_i(t+1)] \end{aligned} \tag{A 22}$$

Substituting system dynamics (A 4) into $\bar{V}_i(t+1)$ in (A 22), one obtains

$$\begin{aligned} V_i(t) &= \delta x_i^{\mathrm{T}}(t)D_{1i}(t)\delta x_i(t) + \delta u_i^{\mathrm{T}}(t)D_{2i}(t)\delta u_i(t) + \delta z_i^{\mathrm{T}}(t)D_{3i}(t)\delta z_i(t) \\ &\quad + \delta u_i^{\mathrm{T}}(t)D_{4i}(t)\delta x_i(t) + \delta z_i^{\mathrm{T}}(t)D_{5i}(t)\delta x_i(t) + \delta u_i^{\mathrm{T}}(t)D_{6i}(t)\delta z_i(t) \\ &\quad + d_{1i}^{\mathrm{T}}(t)\delta x_i(t) + d_{2i}^{\mathrm{T}}(t)\delta u_i(t) + d_{3i}^{\mathrm{T}}(t)\delta z_i(t) + \text{constant}, \end{aligned} \tag{A 23}$$

where coefficients are updated according to (A 6) to (A 14) with T replaced by $t+1$ and $T-1$ by t. The variational feedback control coefficients $\alpha_{1i}(t)$, $\beta_{1i}(t)$, $\alpha_{2i}(t)$ and $\beta_{2i}(t)$ are obtained following the derivations of (A 15) to (A 18). At the convergence of the high-level algorithm, $\delta u_i(t)$ can be expressed in terms of $\delta x_i(t)$ and $\delta z_i(t)$ by using the first-order necessary condition (A 15):

$$\delta u_i(t) = \alpha_i(t) + \beta_i(t)\delta x_i(t) + \gamma_i(t)\delta z_i(t) \tag{A 24}$$

Thus, global variational feedback controls are obtained.

Appendix B

Proof of Theorem 3.1: The main idea to prove Theorem 3.1 is to show that the variational feedback control coefficients obtained in Appendix A satisfy the

discrete Riccati equation for optimal control (Kwakernaak and Sivan 1972). To simplify the proof, stack control variables and corresponding matrices are formed to re-derive the backward sweep of DDP by following the procedure of Appendix A:

$$u_{si}(t) = \begin{bmatrix} u_i(t) \\ z_i(t) \end{bmatrix} \tag{B 1}$$

$$B_{si}(t) = [B_i(t),\ C_i(t)] \tag{B 2}$$

$$R_{si}(t) = \begin{bmatrix} R_i(t) & 0 \\ 0 & S_i(t) \end{bmatrix} \tag{B 3}$$

The variational cost-to-go at stage t is

$$\begin{aligned} V_i(t) &= g_{it}(x_i(t),\ u_{si}(t)) + [0,\ \lambda^{\mathrm{T}}(t)]x_{si}(t) - \sum_{\substack{j=1 \\ j\neq i}}^{N} [\lambda_j^{\mathrm{T}} M_{ji}(t)x_i(t)] + \bar{V}_i(t+1) \\ &= \delta x_i^{\mathrm{T}}(t) D_{1i}(t)\delta x_i(t) + \delta u_{si}^{\mathrm{T}}(t) D_{2i}(t)\delta u_{si}(t) + \delta u_{si}^{\mathrm{T}}(t) D_{4i}(t)\delta x_i(t) \\ &\quad + d_{1i}^{\mathrm{T}}(t)\delta x_i(t) + d_{2i}^{\mathrm{T}}(t)\delta u_{si}(t) \end{aligned} \tag{B 4}$$

From the first-order necessary condition, the optimal variational feedback controls are obtained as:

$$\delta u_{si}(t) = \alpha_{si}(t) + \beta_{si}(t)\delta x_i(t) \tag{B 5}$$

where

$$\alpha_{si}(t) = -\tfrac{1}{2} D_{2i}^{-1}(t) d_{2i}(t) \tag{B 6}$$

$$\beta_{si}(t) = -\tfrac{1}{2} D_{2i}^{-1}(t) D_{4i}(t) \tag{B 7}$$

Substitute (B 5) into (B 4), then the optimal variational cost-to-go is

$$\bar{V}_i(t) = \delta x_i^{\mathrm{T}}(t) E_i(t)\delta x_i(t) + \mu_i^{\mathrm{T}}(t)\delta x_i(t) + \text{constant} \tag{B 8}$$

where

$$E_i(t) = D_{1i}(t) + \beta_i^{\mathrm{T}}(t) D_{2i}(t)\beta_i(t) + \beta_i^{\mathrm{T}}(t) D_{4i}(t) \tag{B 9}$$

It is easy to see that $D_{2i}(t)$ is positive definite since $R_i(t)$ and $S_i(t)$ are positive definite. Therefore, $E_i(t)$ is symmetric and positive semidefinite. The matrices $D_{1i}(t)$, $D_{2i}(t)$ and $D_{4i}(t)$ can be obtained similar to (A 6), (A 7), and (A 9):

$$D_{1i}(t) = Q_i(t) + A_i^{\mathrm{T}}(t) E_i(t+1) A_i(t) \tag{B 10}$$

$$D_{2i}(t) = R_i(t) + B_i^{\mathrm{T}}(t) E_i(t+1) B_i(t) \tag{B 11}$$

$$D_{4i}(t) = 2\beta_{si}^{\mathrm{T}}(t) E_i(t) A_i^{\mathrm{T}}(t) \tag{B 12}$$

When the DDP algorithm converges, the nominal state and control trajectories are optimal. Furthermore,

$$\alpha_{si}(t) \equiv 0 \tag{B 13}$$

causing variational controls and variational states to vanish. By substituting

equations (B 10), (B 11) and (B 12) into (B 7) and (B 9), the recursive form of the optimal variational feedback coefficients are obtained as:

$$\beta_{si}(t) = -[B_{si}(t)^{\mathrm{T}}(t)E_i(t+1)B_{si}(t) + R_{si}(t)]^{-1}B_{si}^{\mathrm{T}}(t)E_i(t+1)A_i(t) \quad \text{(B 14)}$$

with

$$E_i(t) = A_i^{\mathrm{T}}(t)E_i(t+1)[A_i(t) + B_{si}(t)\beta_{si}(t)] + Q_i(t), \quad t = 0, 1, \ldots, T-1 \quad \text{(B 15)}$$

and

$$E_i(T) = Q_i(T)$$

Note that (B 14) and (B 15) have the equivalent form to the discrete Riccati equation for optimal control (Kwakernaak and Sivan 1972, pp. 494). Therefore, the variational feedback control

$$\delta u_{si}(t) = \beta_{si}(t)\delta x_i(t) \quad \text{(B 17)}$$

is optimal and asymptotically stable (Caine and Mayne 1970, Kwakernaak and Sivan 1972). □

Appendix C

Proof of Theorem 3.2: To prove Theorem 3.2, let us first consider the non-linear programming problem P' as follows:

(P'):

$$\min_{y_s} [\tfrac{1}{2}y_s^{\mathrm{T}}G_s y_s + h_s^{\mathrm{T}}y_s + \text{constant}] \quad \text{(C 1)}$$

subject to

$$F_s y_s = b_s \quad \text{(C 2)}$$

Suppose G_s is positive and F_s is of full rank. Define the lagrangian as follows:

$$L \equiv \tfrac{1}{2}y_s^{\mathrm{T}}G_s y_s + h_s^{\mathrm{T}}y_s + \lambda^{\mathrm{T}}(F_s y_s - b_s) \quad \text{(C 3)}$$

then the dual function is given by

$$\Phi(\lambda_s) = \min_{y_s} L \quad \text{(C 4)}$$

Proposition C.1: *For problem P', the dual function $\Phi(\lambda_s)$ is of the following quadratic form:*

$$\Phi(\lambda_s) = \lambda_s^{\mathrm{T}}\Gamma\lambda_s + \zeta^{\mathrm{T}}\lambda_s + \text{constant} \quad \text{(C 5)}$$

where Γ is negative definite.

Proof: To minimize (C 3), the following necessary condition must be satisfied:

$$G_s y_s + h + F_s^{\mathrm{T}}\lambda_s = 0 \quad \text{(C 6)}$$

The optimal y_s is thus given by

$$y_s^* = -G_s^{-1}(h + F_s u^T \lambda_s) \tag{C 7}$$

Substituting (C 7) into (C 3), one obtains (C 5) with

$$\Gamma = -\tfrac{1}{2} F_s G_s^{-1} F_s^T$$
$$\zeta^T = -(h_s^T G_s F_s^T + b^T)$$

Since G_s is positive definite F_s is of full rank, Γ is negative definite. □

It will be shown that under the LQ assumption problem P can be converted to P'. To do so, consider problem P'' which is equivalent to problem P under the LQ assumption:

(P''):

$$\min_{\substack{u(t)\\ z(t)}} \tfrac{1}{2}\Bigg[\sum_{t=0}^{T-1} [x^T(t)Q(t)x(t) + u^T(t)R(t)u(t) + z^T(t)S(t)z(t)] + x^T(T)Q(T)x(T)\Bigg] \tag{C 8}$$

subject to

$$x(t+1) = A(t)x(t) + B(t)u(t) + C(t)z(t) \tag{C 9}$$

and

$$z(t) = M(t)x(t), \quad t = 0, 1, \ldots, T-1 \tag{C 10}$$

where $Q(\cdot)$ is positive semidefinite, $R(\cdot)$ and $S(\cdot)$ are positive definite. The matrices $Q(t)$, $R(t)$, $S(t)$ and $M(t)$ have the following structures:

$$C(t) = \text{diag}[C_1(t), \ldots, C_N(t)]$$
$$Q(t) = \text{diag}[Q_1(t), \ldots, Q_N(t)]$$
$$R(t) = \text{diag}[R_1(t), \ldots, R_N(t)]$$
$$S(t) = \text{diag}[S_1(t), \ldots, S_N(t)]$$

$$M((t) = \begin{bmatrix} 0 & M_{12}(t) & \ldots & M_{1N}(t) \\ M_{21} & 0 & \ldots & M_{2N}(t) \\ . & . & \ldots & . \\ M_N 1(t) & M_{N2}(t) & \ldots & 0 \end{bmatrix}$$

It will be shown that P'' is equivalent to P' by showing that the object function and constraints of P'' can be represented in the form of P'. First consider the constraints of P''. Define for convenience a stack form of control and coupling variables as

$$v_t \equiv [u^T(0), z^T(0), \ldots, u^T(t), z^T(t)]^T \tag{C 12}$$

For linear systems, it is known that the state variable $x(t)$ can be expressed as a linear combination of inputs and initial state, i.e.,

$$x(t) = \Theta_t v_{t-1} + \Xi_t x(0) = \Theta_t v_{t-1} + \zeta_t \tag{C 13}$$

Substitute (C 13) into the coupling equation (C 10) and note

$$v_t = [v_{t-1}^T, u^T(t), z^T(t)]^T \tag{C 14}$$

then one obtains

$$F_t v_t = b_t, \quad t = 0, \ldots, T-1 \tag{C 15}$$

where

$$F_t = [-M(t)\Theta_t, 0, I] \tag{C 16}$$

$$b_t = M(t)\zeta_t, \quad t = 1, \ldots, T-1 \tag{C 17}$$

$$F_0 = [0, I] \tag{C 18}$$

$$\xi_0 = x(0) \tag{C 19}$$

Note that F_t is of full rank because of the unit matrix in (C 16). Let

$$y_s \equiv v_{T-1} \tag{C 20}$$

then v_t is the first $(t+1)(r+m)$ components of y_s. That is,

$$v_t = Z_t y_s \tag{C 21}$$

where

$$Z_t = [I_t, 0] \quad \text{and} \tag{C 22}$$

I_t is a unit matrix with the proper dimension. Again, Z_t is of full rank. Equation (C 15) is equivalent to constraint (C 2) with

$$F_s = \begin{bmatrix} F_0 Z_0 \\ F_1 Z_1 \\ \vdots \\ F_{T-1} Z_{T-1} \end{bmatrix} \tag{C 23}$$

Also F_s is of full rank because $F_t Z_t$ is of full rank for $t = 0, \ldots, T-1$.

It can be shown that the objective function of P'' in (C 8) can be rewritten in the form of (C 1) with

$$G_s = R_s + \Omega_s \tag{C 24}$$

where

$$R_s = \text{diag}[R(0), S(0), \ldots, R(T-1), S(T-1)] \tag{C 24}$$

$$\Omega_s = Z_0^T \Theta_1^T Q(1) \Theta_1 Z_0 + \cdots + Z_{T-1}^T \Theta_T^T Q(T) \Theta_T Z_{T-1} \tag{C 25}$$

Since $R(t)$ and $S(t)$ are positive definite and $Q(t)$ is positive semidefinite, R_s is positive definite and Ω_s is positive semidefinite. Consequently, G_s is positive definite. P'' is therefore equivalent to problem P'. By using Preposition C.1, Theorem 3.2 is thus proved. □

Appendix D

Object functions and system dynamics for numerical testing

D.1. *Problem* 1

The first problem for numerical testing is an LQ problem with 11 subsystems. The stage-wise cost functions and system dynamics are given in (3.6) and (A 4), respectively. The matrices in these equations are given below:

i 1 2 3

$$A_i \quad \begin{bmatrix} -1 & -0{\cdot}2 \\ 0{\cdot}1 & -2 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ -0 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0{\cdot}5 & 0 \\ 0 & -1 & 0 \\ 0 & 0{\cdot}5 & -1 \end{bmatrix}$$

i 4 5 6

$$A_i \quad \begin{bmatrix} 0{\cdot}9 & 0{\cdot}2 & 0 \\ 0{\cdot}1 & 0{\cdot}1 & 0 \\ 0 & 0{\cdot}5 & 1 \end{bmatrix}, \begin{bmatrix} 1{\cdot}5 & 0 & 0 \\ -0{\cdot}5 & -1 & 0 \\ 0 & 0{\cdot}5 & -1 \end{bmatrix}, \begin{bmatrix} 2 & 0 & 0 \\ -1 & -1 & 0 \\ 0 & 0{\cdot}5 & -1 \end{bmatrix}$$

i 7 8 9

$$A_i \quad \begin{bmatrix} 0{\cdot}5 & 0 & 0 \\ 0{\cdot}1 & -1{\cdot}2 & 0 \\ 0 & 0{\cdot}5 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ -0{\cdot}4 & -0{\cdot}75 & 0 \\ 0 & 0{\cdot}5 & -1 \end{bmatrix}, \begin{bmatrix} 0 & -1 & 0 \\ 1 & -2 & 0 \\ 0 & 0{\cdot}5 & -1 \end{bmatrix}$$

i 10 11

$$A_i \quad \begin{bmatrix} -1 & 0{\cdot}5 & 0 \\ 1 & -1 & 0 \\ 0 & -0{\cdot}5 & -1 \end{bmatrix}, \begin{bmatrix} -3 & 0{\cdot}6 & 0 \\ 0{\cdot}2 & -0{\cdot}5 & 0 \\ 0 & -0{\cdot}5 & 0{\cdot}8 \end{bmatrix}$$

$$B_1 = B_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad B_i = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad i = 3, 4, \ldots, 11$$

$$C_1 = \begin{bmatrix} 0{\cdot}5 \\ 0 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 0{\cdot}2 \\ 0 \end{bmatrix}, \quad C_i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad i = 3, 4, \ldots, 11$$

$$z_1(t) \equiv [1\ 0]x_2(t) + \sum_{j=3}^{11} [1\ 0\ 0]x_j(t), \quad z_2(t) \equiv [1\ 0]x_1(t),$$

$$z_i(t) \equiv 0, \ i = 3, 4, \ldots, 11$$

$$Q_1 = Q_2 = 2I_2, \quad Q_i = 2I_3, \ i = 3, 4, \ldots, 11$$

where I_i is an i by i unit matrix,

$$R_i = 1 \quad i = 1, 2, \ldots, 11$$

$$S_1 = S_2 = 1$$

D.2. *Problem* 2

The second problem tested is identical with Problem 1 except that quartic terms are added to the quadratic object function. The stage-wise object functions are:

$$g_{it}(x_i(t), u_i(t), z_i(t)) = \tfrac{1}{2}\{x_i^{\mathrm{T}}(t)Q_i x_i(t) + u_i^{\mathrm{T}}(t)R_i u_i(t) + z_i^{\mathrm{T}}(t)S(t)\ z_i(t)] + 0{\cdot}25x_{i2}^4(t), \quad i = 1, 2 \tag{D 4}$$

$$g_{ii}(x_i(t), u_i(t), z_i(t)) = \tfrac{1}{2}\{x_i^{\mathrm{T}}(t)Q_i x_i(t) + u_i^{\mathrm{T}}(t)R_i u_i(t) + z_i^{\mathrm{T}}(t)S(t)\, z_i(t)] + 0{\cdot}25x_{i3}^4(t), \quad i = 3, 4, \ldots, 11 \qquad \text{(D 5)}$$

References

Bertsekas, D. P., and Tsitsiklis, J. N., 1989, *Parallel and Distributed Computation* (Englewood Cliffs, NJ: Prentice Hall).

Bryson, A. E. Jr., and Ho, Y. C., 1975, *Applied Optimal Control*, revised printing (Hemisphere Publishing).

Caine, P. E., and Mayne, D. Q., 'On the discrete time matrix Riccati equation of optimal control'. *International Journal of Control*, **12**, 785–794.

Chang, S. C., Chang, T. S., and Luh, P. B., 1989 A hierarchical decomposition for large scale optimal control problems with parallel processing structure. *Automatica*, **25**, 77–86.

Jamshidi, M., 1983, *Large Scale Systems* (Amsterdam: North-Holland).

Kwakernaak, H., and Sivan, R., 1972, *Linear Optimal Control Systems* (Wiley-Interscience).

Loostma, R. A., and Ragsdell, K. M. State-of-the-art in parallel nonlinear optimization. *Parallel Computing*, **6**, 133–155.

Leunberger, D. G., 1984, *Linear and Nonlinear Programming*, second edition (Addison Wesley).

Pearson, J. D., 1971, Dynamic decomposition techniques. *Optimization Methods for Large Scale Systems*, D. A. Wismer (editor) (New York: McGraw-Hill).

Perry, P. F., 1984, Spatial and time decomposition algorithms for dynamic nonlinear network optimization using duality. *Journal of Optimization Theory and Applications*, **42**, 77–101.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., 1988, *Numerical Recipes in C* (Cambridge, UK: Cambridge University Press).

Singh, M. G., 1980, *Dynamical Hierarchical Control* (Amsterdam: North-Holland).

Straeter, T. A., 1975, A parallel variable metric optimization algorithm. *NASA Technical Note*, NASA TN D-7329.

Tang, J., Luh, P. B., and Change, T. S., 1991, A parallel algorithm for long horizon optimal control problems using the mixed coordination method. *International Journal of Control*, **53**, 1395–1412.

Van Laarhoven, P., Parallel variable metric algorithms for unconstrained optimization. *Mathematical Programming*, **33**, 68–81.

Yakowits, S., and Rutherford, B., 1984, Computational aspects of discrete-time optimal control. *Applied Mathematics and Computation*, **15**, 29–45.