2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Dynamic Processor Allocation for Multiple RHC Systems in Multi-Core Computing Environments

Ali Azimi and Brandon W. Gordon

Abstract—**This paper develops a new dynamic processor allocation algorithm for multiple receding horizon controllers (RHC) executing on a multi-core parallel computer. The proposed formulation accounts for bounded model uncertainty, sensor noise, and computation delay. A cost function appropriate for control of multiple coupled vehicle systems on multiple processors is used and an upper bound on the cost as a function of the execution horizon is employed. A parallel processing adaptation of the SNOPT optimization package is used and the efficiency factor of the parallel optimization routine is estimated through simulation benchmarks. Minimization of the cost function upper bound combined with the efficiency factor information results in a combinatorial optimization problem for dynamically allocating the optimal number of logical processors for each RHC subsystem. The new approach is illustrated through simulation of a leader-follower control system for two 3DOF helicopters running on a computer with two quad-core processors.**

I. Introduction

RECEDING horizon control (RHC) is a repeated online solution of a finite horizon open-loop optimal control problem [1]. Application of RHC to control problems with multiple subsystems is addressed by applying RHC to the individual subsystems while the information regarding the state variables, or trajectories, are exchanged between them, which leads to a decentralized formulation. In most of the decentralized architectures, each subsystem is optimized on a single computer. Besides, some decentralized RHC approaches optimize a group of subsystems on a single computer (see [12] for example). In this case, multiple RHC processes must be scheduled in an appropriate manner on a single processor and it was discussed in [9], [10], and [11]. In those approaches, the execution horizons of all subsystems were selected by solving an optimization problem.

However, most new computer designs are adopting a multi-core architecture where multiple logical processors running in parallel are contained on each physical processor package on the computer. This allows an increase in processing speed, with significantly improved performance than networked computing, provided appropriate algorithms are available to take advantage of the additional logical processors (cores). It is anticipated by computing processor manufactures that in the near future computers composed of hundreds of cores will be available. However, in order to benefit from such architectures, a systematic approach for performing the computations in parallel and dynamically allocating the processors becomes necessary. For the case of RHC which is very computationally expensive, the performance of RHC can be significantly improved using more processors through reduction of the execution horizon and the computation delay. In addition, more complex and accurate models will be possible as well as examining multiple scenarios for more globally optimal results.

In addition, multi-core architectures are potentially superior to distributed/network computing, since there are much smaller communication delays/latencies and higher bandwidth than even gigabit class networks such as gigabit Ethernet. This enables the application of multi-core computers to RHC problems that would not be possible on a computational network, especially when using serial algorithms such as sequential quadratic programming (SQP), which are currently used to solve most types of RHC optimization problems.

Some attempts have been performed to apply optimal control problems on parallel computers, including the approach presented in [3] using dynamic programming and a space decomposition scheme in which the global optimal control problem is reduced to the optimization of sub-problems. In [4] and [5], parallel algorithms are presented for optimal control problems with long prediction horizon using time decomposition techniques. A two-phase parallel computing method is presented in [6] to obtain the solution of receding horizon controller for constrained nonlinear systems. The approach in this paper is distinguished from these existing approaches in that it considers dynamic allocation of the processors in response to changing uncertainty and disturbances of the RHC subsystems.

In this paper multiple RHC subsystems are considered on a single computer with multiple processors. However, the number of processors assigned to each subsystem is changing and is determined by a proposed algorithm. The new technique determines the execution horizons and allocates the appropriate number of processors for all subsystems. The execution horizon determination of each subsystem and processor allocation while optimizing the performance is cast into a constrained optimization problem. Online solution of the foresaid optimization problem using the updated optimization parameters, results in dynamically determining the processor allocation.

A. Azimi and B. W. Gordon are with Department of Mechanical and Industrial Engineering, Concordia University, 1455 de Maisonneuve Blvd. West, Montreal, Quebec, H3G 1M8, CANADA (phone: (514) 848-2424 ext. 7058; fax: (514) 848-3175; (e-mail: a_azi@encs.concordia.ca, bwgordon@encs.concordia.ca).

978-1-4244-4524-0/09/$25.00 ©2009 AACC

II. Problem Statement

A. Decentralized RHC formulation

Consider N dynamically decoupled subsystems using the RHC approach. The subsystems are supposed to have connection with each other by exchanging their information. The term *system* used in this paper, refers to all subsystems. Furthermore, consider the following nominal equation for the i^{th} subsystem:

$$\dot{\mathbf{x}}_i = \mathbf{f}_i\left(\mathbf{x}_i(t), \mathbf{u}_i(t), t\right), \qquad i = 1, ..., N \tag{1}$$

which serves as a model for the actual subsystem described by:

$$\dot{\hat{\mathbf{x}}}_i = \mathbf{f}_i\left(\hat{\mathbf{x}}_i(t), \mathbf{u}_i(t), t\right) + \mathbf{g}_i(\hat{\mathbf{x}}_i, \mathbf{u}_i, t), \; i = 1, \ldots, N \tag{2}$$

where $\mathbf{x}_i(t) \in \Re^{p_i}$ and $\hat{\mathbf{x}}_i(t) \in \Re^{p_i}$ are the nominal and actual states of the i^{th} subsystem, respectively. The input vector $\mathbf{u}_i(t) \in \Re^{m_i}$ satisfies the constraints $\mathbf{u}_i(t) \in U_i$ ($\forall t \geq 0$), where U_i is the allowable set of inputs for subsystem i. Furthermore, it is assumed that (A1-A3) in [2] are also satisfied; that is, $\mathbf{f}_i$ is twice differentiable, U_i is compact and convex, and subsystem (1) has a unique solution for a given initial condition.

Definition 1. The set A_i is called the neighbouring set of subsystem i, and consists of any subsystem that its information is used in the control of subsystem i.

The finite horizon cost associated to i^{th} subsystem is defined as follows [11]:

$$J_i\left(t, T_i, \mathbf{x}_i, \mathbf{u}_i, \tilde{\mathbf{x}}_i\right) = V_i\left(\mathbf{x}_i\left(T_i + t; t\right)\right) + \tag{3}$$

$$\int_t^{t+T_i} \left(q_i\left(\mathbf{x}_i(\tau; t), \mathbf{u}_i(\tau)\right) + \sum_{j \in A_i} g_{ij}\left(\mathbf{x}_i(\tau; t), \mathbf{x}_j(\tau; t)\right) \right) d\tau$$

where $\tilde{\mathbf{x}}_i$ is a vector containing the states of all neighbours of the i^{th} subsystem. g_{ij} is a function which defines the interaction between two nodes of the system $\mathbf{x}_i$ and $\mathbf{x}_j$. T_i is the optimization horizon of the RHC controller associated to i^{th} subsystem. $V_i(.)$ defines the final penalty and $\mathbf{x}_i(\tau; t)$ is the state vector of i^{th} subsystem at time τ which has been sampled at time t. The optimal cost is then given by [8]:

$$J_i^*\left(t, T_i, \mathbf{x}_i^*, \mathbf{u}_i^*, \tilde{\mathbf{x}}_i\right) = \inf_{\mathbf{u}_i(\cdot)} J_i\left(t, T_i, \mathbf{x}_i, \mathbf{u}_i, \tilde{\mathbf{x}}_i\right) \tag{4}$$

The optimized trajectory resulting from (4) is defined as $(\mathbf{x}_{T,i}^*(\tau; t), \mathbf{u}_{T,i}^*(\tau; t))$, $\tau \in (t, t + T_i]$. In the closed loop RHC the calculated input $\mathbf{u}_{T,i}^*(\tau; t)$ is applied to the actual subsystem (2), while $\tau \in (t, t + \delta_i]$ and δ_i is called the *Execution Horizon* of subsystem i ($\delta_i < T_i$).

B. Real-time processor allocation

Assuming the control of N subsystems is desired on a computer with m_p identical physical processors, such that each physical processor j has m_j logical processors. The problem under study is processor allocation, which is defining the appropriate number of logical processors and its computation topology to each subsystem. The term *computation topology* is defined later.

Definition 2. A *logical processor* is referred to as the smallest independent processing unit. Multiple logical processors are used to form *physical processors*. Multiple physical processors can be put together in a *computer*.

For RHC control of a single apparatus using a single computer, the calculated inputs are applied for a period equal to the execution horizon of the receding horizon controller. Therefore, in real-time implementation, a common way is to define a real-time periodic task [7], with its period equal to the execution horizon of that RHC. Assume the subsystems described in (1) and (2) are connected to a set of multiple computers for feedback control, using RHC method, as explained earlier. From a computer control point of view, each control system can be handled as a periodic task in the real-time programming. The period of each periodic task is equal to the *execution horizon* of its related subsystem. Determining these periods (or execution horizons) is not trivial. In this paper, a systematic approach is presented to calculate these periods. Based on that, the appropriate processing units, i.e. logical processors, are assigned to each subsystem.

III. Implementation of RHC on Multi-Core Processors

Assume a single task, which is implementation of a single RHC system, needs to be computed on a computer with more than one identical multi-core processor. In addition, assume the computation time c_i is known if this task is computed on one of the logical processors only. However, if n_i identical logical processors are used, the computation time should be decreased and the minimum computation time is c_i / n_i in the ideal case. However, depending on the type of the task that needs to be parallelized and the method used in the parallelization, the actual computation time, i.e. c_i', is more than the ideal case in practice. An efficiency factor, η_i, is introduced as follows for identical logical processors:

$$\eta_i = c_i / \left(n_i c_i'\right) \tag{5}$$

where $\eta_i \in (0, 1]$. It should be noted that if all of the n_i logical processors, used in the calculation of the foresaid task, are not on a single processor, the efficiency factor may vary depending on the distribution of cores on different processors, which shall be discussed later.

In the following, the method used in calculation of RHC on parallel processors is explained. Implementing RHC on parallel processing units deals with dividing the associated nonlinear optimization problem to sub-problems and calculating them on different units.

A. Parallel SQP implementation for a single RHC

Sequential quadratic programming (SQP) methods are among the most effective nonlinear programming algorithms for solving differentiable nonlinear optimization problems

[13]. A Fortran implementation of a SQP algorithm is presented in [13] for parallel processors. Part of this idea corresponding to the Jacobian and cost function calculation, is employed in this paper to apply RHC on parallel processors.

In order to solve the optimization problem associated with RHC and to find the appropriate inputs corresponding to its prediction horizon, the SNOPT optimization package [14] is used. To formulate the problem, every input is estimated by a cubic spline. Thereby, the aim of optimization is finding the spline coefficients of all inputs.

In order to calculate the cost function based on the optimization variables, the following steps should be done:

Step 1- Run the spline routine to find the inputs based on the spline coefficients

Step 2- Find the state trajectories by simulating the subsystem from the initial conditions and using the calculated inputs in Step 1.

Step 3- Calculate the cost by using the input and state trajectories obtained in the previous steps. It should be noted that, this cost calculation includes estimating an integral using trapezoidal integration.

Therefore, the RHC optimization is time consuming due to complexity of its cost function calculation. Besides, the gradient of the cost cannot normally be calculated analytically, and it is typically computed using center finite difference approximations.

A simple yet effective approach for implementing this optimization on parallel processors, is to calculate the gradient (or Jacobian) and the cost function in parallel. However, in this paper only the Jacobian is calculated in parallel. Using this approach, a relatively high efficiency factor of 0.9 is obtained in the example section when two logical processors were used.

Remark 1. It should be emphasized that the processor allocation approach presented in this paper, works with any efficient parallel processing optimization method. However, the method should be flexible enough to work when the number of processors used in each subsystem varies dynamically.

B. Efficiency factor calculation

As mentioned earlier, the number of logical processors used in calculating a particular task affects the efficiency factor. However, there are also some other factors that affect it and are discussed presently.

The way the logical processors access memory, can dramatically affect the performance of the parallel algorithms, and consequently affect the efficiency factor. Particularly, by using Non-Uniform Memory Architecture (NUMA), separate memory is assigned to each logical processor, which avoids the performance hit when some processors try to access the same memory. NUMA can be considered as a tightly form of cluster computing, and is used in the computers performing the simulations in this paper. Another important factor is the available cache memory (L1 and L2) for each processor.

Remark 2. The computer used in the example section has two quad-core Intel Xeon 5300 series processors. Each processor has two dies, each includes two cores. Its L1 cache has 32KB for instructions and 32KB for data. In addition, it has 4MB L2 cache per die that means 8MB L2 cache for each physical processor [15].

Based on the foresaid factors, in addition to the number of logical processors, the distribution of them on the physical processors, also affects the efficiency factor. This is referred to as *computation topology* in this paper.

As an example for possibility of having deferent computation topologies, assume 4 tasks are operated on a computer with two quad-core processors, while based on the computational need of them, tasks 1 to 4 require 3, 2, 2, and 1 core, respectively. Two different architectures are presented in Fig. (1-a) and Fig. (1-b).

Remark 3. It should be noted that the efficiency factor may still vary even with fixed logical processors and computation topology. For example, in calculation of an optimization problem, the number of iterations may vary that might cause slightly different efficiency factor. However, in this paper, it is assumed that the efficiency factor is only depends on the number of logical processors and the computation topology, as long as the simulations are done on a same computer with the same parallel algorithm.

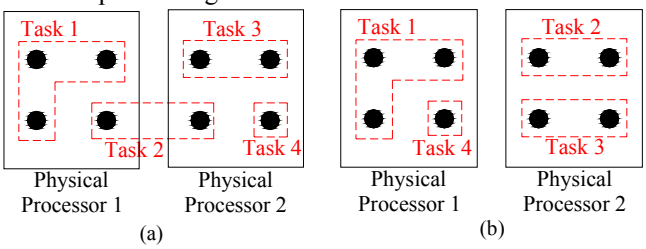

Fig. 1. Different architectures assigned to the set of 4 tasks in two quad-core processors. In (b) each processor has 2 tasks and there is no need to have data exchange between processors.

Based on the foresaid discussions and the relation presented in (5), the efficiency factor, regarding the calculation of a single RHC, can be obtained using the following procedure:

Step 1- Perform the simulation using only a single logical processor and obtain the computation time c_i.

Step 2- Perform the same simulation using more than one processor and obtain c_i' for different number of logical processors and every possible computation topologies.

Step 3- Find the efficiency factor for all cases using (5) and store the results in a lookup table.

Remark 4. The foresaid procedure should be implemented several times using different initial conditions and different possible problem parameters, and the final efficiency factor is the average of them. This task is done offline and the values of efficiency factors are used later in the dynamic processor allocation algorithm.

IV. Dynamic Processor Allocation Algorithm

For the proposed approach, the execution horizons of all subsystems should be defined such that the overall performance of the system is maximized. However, the execution horizon is dependent on the number of logical processors assigned to it, the efficiency factor, and the computation time of the task that will be discussed later in this section. Similar to the approach presented in [11], in order to evaluate the performance of the system, the following cost function is proposed as the cost of the closed loop system from time t to $t+T_a$, where T_a is the period that calculated execution horizons would be applied to:

$$\hat{J}_a = \sum_{i=1}^{N} \left(\int_t^{t+T_a} \left(\begin{array}{l} q_i\left(\hat{\mathbf{x}}_i(\tau), \mathbf{u}^*_{T,i}\left(\tau; t_k^i\right)\right) + \\ \sum_{j\in A_i} g_{ij}\left(\hat{\mathbf{x}}_i(\tau), \mathbf{x}^*_{T,j}\left(\tau; t_k^j\right)\right) \end{array} \right) d\tau \right) \quad (6)$$

where $t_k^i = t + (k-1)\delta_i$ and $\mathbf{u}^*_{T,i}\left(\tau; t_k^i\right)$ is the optimal input applied to subsystem i. In addition, the following is presented as the estimation of $\hat{J}_a$ [11]:

$$\bar{J}_a = \sum_{i=1}^{N} \left(\frac{T_a}{\delta_i} \int_{t_i}^{t_i+\delta_i} \left(\begin{array}{l} q_i\left(\hat{\mathbf{x}}_i(\tau), \mathbf{u}^*_{T,i}\left(\tau; t_{s,i}\right)\right) + \\ \sum_{j\in A_i} g_{ij}\left(\hat{\mathbf{x}}_i(\tau), \mathbf{x}^*_{T,j}\left(\tau; t_{s,j}\right)\right) \end{array} \right) d\tau \right) \quad (7)$$

where $t \le t_i$ and $t > t_{s,i}$. t_i represents the time that new inputs are applied to subsystem i based on the sampled data at $t_{s,i}$ and t indicates the start time of dynamic allocator. In the following, an upper bound on (7) is presented and it shall be used to find the optimal number of logical processors for each subsystem.

In the Lemma 4 of [11], by considering some common assumptions, the following upper bound was presented on the $\bar{J}_a$:

$$\begin{aligned} \bar{J}_a &\le \sum_{i=1}^{N} G_i(\delta_i) = \\ &\sum_{i=1}^{N} \left(\frac{T_a}{\delta_i} \left(\begin{array}{l} \int_{t_i}^{t_i+\delta_i} q_i\left(\mathbf{x}^*_{T,i}\left(\tau; t_{s,i}\right), \mathbf{u}^*_{T,i}\left(\tau; t_{s,i}\right)\right) d\tau \\ + P_i\left(B_i^{(1)}(\delta_i) + B_i^{(2)}(\delta_i)\right) \end{array} \right) \right) \\ &+ \sum_{i=1}^{N} \left(\frac{T_a}{\delta_i} \sum_{j\in A_i} \left(\begin{array}{l} \int_{t_i}^{t_i+\delta_i} g_{ij}\left(\mathbf{x}^*_{T,i}\left(\tau; t_{s,i}\right), \mathbf{x}^*_{T,j}\left(\tau; t_{s,j}\right)\right) d\tau \\ + L_{ij}^{g}\left(B_i^{(1)}(\delta_i) + B_i^{(2)}(\delta_i)\right) \end{array} \right) \right) \end{aligned} \quad (8)$$

where:

$$\begin{aligned} B_i^{(1)}(\delta_i) &= \frac{b_{s,i}}{L_{x,i}}\left(e^{2L_{x,i}\delta_i} - e^{L_{x,i}\delta_i}\right) + \frac{b_i}{L_{x,i}}\left(\frac{e^{2L_{x,i}\delta_i} - e^{L_{x,i}\delta_i}}{L_{x,i}} - \delta_i\right) \\ B_i^{(2)}(\delta_i) &= \frac{\varepsilon_i L_{u,i}}{\left(L_{x,i}\right)^2}\left(e^{L_{x,i}\left|\delta_i - \delta_i^p\right|} - 1\right)\left(e^{L_{x,i}\delta_i} - 1\right) \end{aligned} \quad (9)$$

In which, $b_{s,i}$ is the state measurement error bound, b_i is the bound on $\mathbf{g}_i(.,.,.)$ in (2), $L_{x,i}$ and $L_{u,i}$ are the Lipschitz constants of $\mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i, t)$ with respect to $\mathbf{x}_i$ and $\mathbf{u}_i$, respectively. ε_i is a constant scalar defined in Corollary 1 of [11], which is used to bound the mismatch between the inputs in two subsequent calculations. δ_i^p is the last prediction horizon of subsystem i, while δ_i is its current value.

In the case of parallel processing, the choice of execution horizon is dependent on the number of logical processors, the efficiency factor and the computation time of the task on a single processor, as mentioned earlier. Assuming the computation time (or worst case computation time) is known, since the efficiency factor is dependent on the number of logical processors and the computation topology, the processor allocation problem cast into finding the optimal number of processors and the computation topology simultaneously. This results in a complex optimization problem.

However, a suboptimal solution can be obtained, by decoupling the problem of calculating the number of processors (n_i) from computation topology assignment. This way, by considering efficiency factor, only as a function of n_i, which referred to as η_i', n_i can be defined. Knowing n_i, the computation topology, and η_i as a result, can be assigned. Finally, δ_i can be calculated based on the calculated η_i and n_i values. This procedure can be explained in the following:

Algorithm 1:

Step 1- Find the efficiency factor as a function of number of cores only (η_i'). This task is done offline and the results are stored in a lookup table.

Step 2- Find δ_i and associated number of logical processors (n_i) by solving the processor allocation optimization problem presented in the following lemma (Lemma 1)

Step 3- Find the optimum computation topology based on the n_i values. This task is done in Lemma 2. Then calculate the actual efficiency factors, η_i, based on n_i and assigned computation topology.

Step 4- Modify δ_i (increase some of them) based on the corrected values of η_i.

Lemma 1:

Suppose the assumptions lead to the upper bound on the $\bar{J}_a$ presented in (8) are all valid. Then the following optimization problem can be used to sub-optimally determine n_i as presented in the Step 2 of Algorithm 1.

$$\begin{aligned} &\min_{n_i} \sum_{i=1}^{N} G_i c_i / \left(n_i \eta_i'\right) \\ &\text{Subject to: } \sum_{i=1}^{N} n_i = \sum_{i=1}^{m_p} m_i \ , \ n_i \ge 1 \end{aligned} \quad (10)$$

where $G_i(.)$ is defined in (8) and δ_i is replaced by its

equivalent $c_i/(n_i\eta_i')$. In addition, c_i is the maximum computation time of i^{th} subsystem on a single core and η_i' in a function of n_i.

Proof:

$G_i(\delta_i)$ presents the upper bound on the performance index of the overall system and minimization of that upper bound should result in minimization of overall system cost, as discussed in [11]. Based on the definition of η_i', the worst case computation time on n_i logical processors is $c_i' = c_i/(n_i\eta_i')$. Moreover, the sampling period of i^{th} subsystem (δ_i) should be more than or equal to c_i' in order to prevent computational overload conditions. In addition, the first constraint ensures that all of the available logical processors are used, while it is assumed that at least one processor is assigned to each subsystem by the use of second constraint. □

Remark 5. The presented problem in (10) is a combinatorial optimization problem because n_i is a positive integer. In the example section, a simple grid search method is used to solve this optimization problem.

A. Computation Topology Assignment

Assume the number of logical processors required for each RHC subsystem is determined. Based on that, the optimal computation topology should be found, which is discussed in the following lemma.

Lemma 2: Assume m_p indicates the total number of physical processors, such that each physical processor j has m_j logical processors; n_i is the number of logical processors for subsystem i, and the total subsystem number is N. x_{ij} is defined as the number of logical processors assigned to subsystem i from physical processor j. In addition, assume that the communication between every two logical processors takes less time if they are from a single physical processor compared to the case that they are from different physical processors. The following maximization problem can optimally define x_{ij}.

$$\max_{x_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{m_p} x_{ij}^{\,2} \tag{11}$$

Subject to:

$$C1: \sum_{j=1}^{m_p} x_{ij} = n_i \ , C2: \sum_{i=1}^{N} x_{ij} \le m_j$$
$$C3: x_{ij} \ge 0,\ i \le N,\ j \le m_p \tag{12}$$

Proof:

Since logical processors need to have communication with each other for every subsystem calculation, it is desired to select them from a single physical processor, if possible. In other way, the communication between every two logical processors placed on different physical processors should be minimized, which can be presented as minimizing the following relation:

$$\sum_{i=1}^{N}\sum_{j=1}^{m_p-1}\sum_{k=j+1}^{m_p} x_{ij}x_{ik} \tag{13}$$

Using the constraint $C1$, the following equality is valid:

$$\sum_{i=1}^{N}\sum_{j=1}^{m_p} x_{ij}^{\,2} = \sum_{i=1}^{N}(n_i)^2 - 2\sum_{i=1}^{N}\sum_{j=1}^{m_p-1}\sum_{k=j+1}^{m_p} x_{ij}x_{ik} \tag{14}$$

Minimization of (13) results in maximization of the left hand side of (14). Moreover, constraint $C1$ ensures that the number of logical processors is assigned correctly while $C2$ indicates that each physical processor is not assigned more than its available logical processors. □

V. Simulation Results

The proposed approach is applied in simulation to the processor allocation of two miniature 3DOF helicopters using RHC scheme, on a computer with 2 quad-core processors. The vehicle dynamics are selected according to [16]. Subsystem 1 is following the trajectory (leader) while subsystem 2, follower, tries to maintain a certain relative position with respect to the leader.

Based on the method in Section III.A, one of the logical processors is assigned for managing the optimization process and the remaining seven processors are used in the parallel processing of the RHC calculations for the two helicopters.

The proposed dynamic processor allocation algorithm, which is presented in Algorithm 1, is compared to the result of the case that 3 and 4 logical processors are assigned to the leader and the follower, respectively. The simulations are performed using Microsoft Visual C++ 2008 and Windows XP. Besides, the uncertainty is added to both subsystems using random variables, such that elements of uncertainty vector $\mathbf{g}_i(.,.,.)$, presented in (2), are selected randomly, while $\|\mathbf{g}_i\| \le b_i$, and b_i is the uncertainty upper bound of subsystem i. b_i is changing based on the pattern shown in Fig. 2. In addition, T_a is selected as 2.0 seconds.

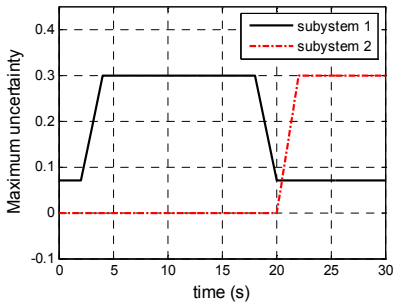

Fig. 2. Changing in the uncertainty upper bound of different subsystems

The paths followed by two subsystems in both cases are presented in Fig. 3 and Fig. 4. Dynamic PA refers to dynamic processor allocation, while Static PA is the other case of having fixed processors for each subsystem. As illustrated in these two figures, in Dynamic RA both subsystems have better performance than Static RA.

It should be noted that the computation topology

assignment is trivial for this example, since there are only two subsystems and two physical processors.

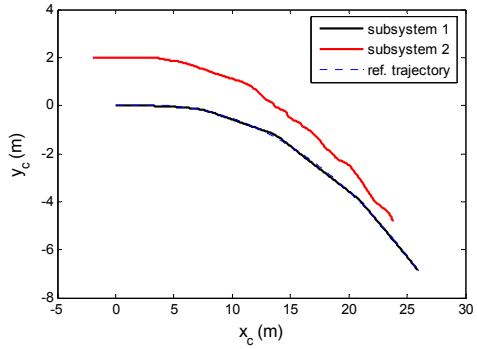

Fig. 3. Paths followed by both subsystems in Dynamic RA case using the proposed algorithm

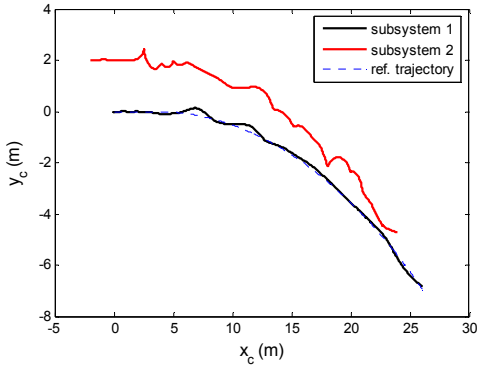

Fig. 4. Paths followed by both subsystems in Static RA case

In addition, the change of processor assignment in the Dynamic RA case is shown in Fig. 5. It should be noted that at the beginning of the Dynamic RA case, the processors are assigned similar to the Static RA case and after 2 seconds, dynamic allocation of processors starts in this example.

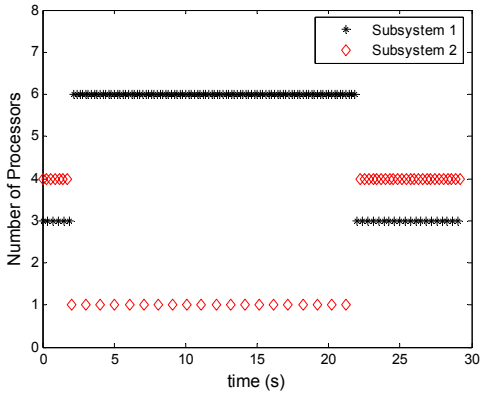

Fig. 5. Number of processors allocated to each subsystem vs. time in the Dynamic RA case

VI. Conclusions

In this paper, a new algorithm for the dynamic processor allocation of multiple receding horizon controllers running on a multi-core computer is developed. In the proposed approach, an execution horizon dependent cost function was used while it accounts for bounded model uncertainty, sensor noise, computation delay, and coupling in the controller cost index. A parallel adaptation of SNOPT optimization package is used by calculating the Jacobian in parallel. By defining the efficiency factor depending on the number of logical processors used to implement each RHC, two combinatorial optimization problems are presented. The first optimization problem, defines the optimal number of logical processors to each RHC, and the second problem assigns the optimal computation topology. Finally, this new approach is illustrated through simulation of two 3DOF helicopters.

Acknowledgment

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for funding this project.

References

[1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality", *Automatica*, 36, 2000, pp. 789-814.

[2] H. Chen, F. Allgower, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability", *Automatica*, 34, 1998, pp.1205-1217.

[3] C. Scheel, B. McInnis, "Parallel processing of optimal control problems by dynamic programming", *Information Sciences*, 25, 1981, pp. 85-114.

[4] T.S. Chang, X.X. Jin, P.B. Luh, X. Miao, "Large scale convex optimal control problems: time decomposition, incentive coordination, and parallel algorithm", *IEEE Trans. Auto. Contr.*, vol. 35, no. 1, 1990, pp. 108-114.

[5] S.C. Chang, T.S. Chang, P.B. Luh, "A hierarchical decomposition for large scale optimal control problems with parallel processing structure", *Automatica*, vol. 25, no. 1, 1989, pp. 77-86.

[6] S.Y. Lin, "A hardware implementable receding horizon controller for constrained nonlinear systems", *IEEE Trans. Auto. Contr.*, vol. 39, no. 9, 1994, pp. 1893-1899.

[7] H. Kopetz, "Real-time systems: Design principles for distributed embedded applications", Chapters 9 and 11, *Springer*, 1997.

[8] T. Keviczky, F. Borrelli, G. J. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems", *Automatica*, 2006

[9] A. Azimi, B.W. Gordon, C.A. Rabbath, "Dynamic scheduling of receding horizon controllers with application to multiple unmanned hovercraft systems", *Proc. of the American Control Conference*, July 2007, pp. 3324-3329.

[10] A. Azimi, B.W. Gordon, C.A. Rabbath, "Dynamic scheduling of decentralized receding horizon controllers on concurrent processors for the cooperative control of unmanned systems", *Proc. of the 46th IEEE Conference on Decision and Control,* Dec., 2007, pp. 518-523.

[11] A. Azimi, B.W. Gordon, C.A. Rabbath, "Dynamic scheduling of multiple decentralized receding horizon controllers subject to computational delay", *Proc. of the American Control Conference*, June 2008.

[12] M. Mercangoz, F.J. Doyle III, "Distributed model predictive control of an experimental four-tank system", Journal of Process Control, vol. 17, 2007, pp. 297-308.

[13] K. Schittkowski, "NLPQLP: A Fortran implementation of sequential quadratic programming algorithm with distributed and non-monotone line search – user's guide, version 2.0", Bayreuth University, Dec. 2004.

[14] P.E. Gill, W. Murray, M.A. Saunders, "User's guide for SNOPT version 7: software for large-scale nonlinear programming", Feb. 2006.

[15] Quad-core Intel Xeon processor 5300 series datasheet, Sept. 2007.

[16] H.A. Izadi, B.W. Gordon, C.A. Rabbath, "Communication bandwidth allocation for decentralized receding horizon control of multiple vehicles", *Proc. of the 2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, China, July 2008, pp. 1195-1200.