Delft University of Technology

Delft Center for Systems and Control

Technical report 07-003

Efficient implementation of serial multi-agent model predictive control by parallelization*

R.R. Negenborn, B. De Schutter, and J. Hellendoorn

If you want to cite this report, please use the following reference instead:
R.R. Negenborn, B. De Schutter, and J. Hellendoorn, "Efficient implementation of serial multi-agent model predictive control by parallelization," *Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control (ICNSC '07)*, London, UK, pp. 175–180, Apr. 2007.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.51.19 (secretary)
fax: +31-15-278.66.79
URL: `http://www.dcsc.tudelft.nl`

*This report can also be downloaded via `http://pub.deschutter.info/abs/07_003.html`

Efficient implementation of serial multi-agent model predictive control by parallelization

R.R. Negenborn, B. De Schutter, J. Hellendoorn

***Abstract*— We discuss an extension of a scheme recently proposed for multi-agent control of large-scale networks, like power networks, road traffic networks, water networks, etc. The original scheme uses serial sequences of agent interactions that under some assumptions make agents locally choose actions that are globally optimal. However, some weaknesses of the approach appear when applied to large-scale networks. We identify these weaknesses and propose, for problems with a tree-structured problem topology, an improvement based on parallelization of the serial scheme. With an example we illustrate and compare the schemes.**

I. Introduction

A. Transportation networks

Our modern society crucially relies on the efficient operation of several types of large-scale transportation networks, like power networks, road traffic networks, water networks, etc. Due to, e.g., high computational requirements, communication delays, or unwillingness to share information, these networks cannot be controlled by a single agent that has access to all actuators and sensors. Instead, a multi-agent or distributed control approach has to be employed, in which the overall network is divided into a number of subnetworks and a control agent is assigned to each subnetwork. Each control agent has to locally determine actions to the actuators in its own subnetwork that give the best overall network performance, using information from sensors in its own subnetwork and communication with other agents [1], [2], [3]. We assume that agents are at least semi-cooperative, in the sense that the agents may have information that they do not want to share with other agents, but that in order to reach desired performance the agents realize that they will have to share some information and be involved in negotiations.

B. Multi-agent model predictive control

Recently, model predictive control (MPC) [4] has been introduced as a strategy for agents to determine their actions in a single-layer multi-agent setting [5]. In an MPC strategy, *at each control cycle*, an agent solves an optimization problem that finds the best local actions to apply to its local subnetwork over a certain prediction horizon under a set of constraints. The agent performs its optimization by making predictions on the evolution of the subnetwork under different sequences of actions and given an initial subnetwork state and constraints on inputs, states, and outputs. The sequence of actions that gives the best performance according to an objective function is determined. The first action of this sequence is implemented, after which the subnetwork evolves to a new state and the next control cycle is started.

For making the predictions within a certain control cycle, each agent uses a model of its subnetwork. Since the physical subnetworks together form the overall network, e.g., due to flows going from one subnetwork to another, the models of the local subnetworks depend on the models of other subnetworks. *Interconnecting variables* are used for modeling these interconnections. A main challenge is how to make individual agents determine values for the interconnecting variables that result in local actions that are globally optimal. Without consistency on the values that the agents assume for the interconnecting variables, the predictions made of the dynamics of a local subnetwork over the prediction horizon will not be accurate, reducing the quality of the control.

C. Parallel versus serial schemes

This challenge can be tackled by having agents perform *iterations of information exchange* between each other, without the intervention of a supervisor, on the values of interconnecting variables [5], [6]. An iteration consists of each agent performing one *step*, involving local computations only. After each agent has performed its step in an iteration, information is exchanged, and the next iteration can be started. One approach for such a scheme is based on a decomposition of an augmented Lagrange formulation of the overall control problem [7]. A typical approach to perform this decomposition is by using an auxiliary problem principle [5], [6]. The resulting scheme is a *parallel* scheme: agents perform local computations simultaneously. An alternative approach to decompose the augmented Lagrange formulation uses a block coordinate descent [7]. This approach has been used before for the unit commitment problem in power systems [8], and has recently been introduced in the context of multi-agent MPC [9]. The block coordinate descent results in a *serial* scheme: one agent at a time performs computations. For small networks, the serial approach has shown preferable properties compared to the parallel approach in terms of decision-making speed and accuracy [9]. In the following we consider extension of this serial approach to larger networks.

D. Parallelized serial schemes

For the scheme that we consider, the local solutions of agents converge over a number of iterations to the overall optimal network solution under a convexity assumption on the overall control problem [7], [9]. Although the local

The authors are with the Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628CD Delft, The Netherlands, {r.r.negenborn,j.hellendoorn}@tudelft.nl, b@deschutter.info. Bart De Schutter is also with the Marine and Transport Technology department of Delft University of Technology.

solutions converge to the global optimum, a disadvantage of this serial approach is that with an increasing number of agents, the number of steps required to complete one single iteration, and thus the total time required for decision making, increases as well. In this paper we examine a method for improving the running time of the serial approach for overall convex problems with a certain tree structure by *parallelizing* the serial scheme. Instead of having one group of agents within which computations are done serially, there may be multiple groups within which computations are done serially simultaneously. We propose to increase the serial decision-making speed by:

- solving fewer local steps by having agents know when their current solution is within some distance from the optimal solution, and by not changing it anymore after this;
- reducing communication between agents by having multiple instances of the serial scheme work simultaneously in smaller groups of agents.

In our approach the agents detect on-line, while solving their subproblems, when the group of agents can be split into smaller groups in which the serial algorithm is performed. This will in particular be beneficial when, e.g., disturbances have only local consequences and not all agents have to be involved in solving these consequences. In these cases, iterations only have to be done by a small number of agents, thus reducing computation and communication requirements, therewith increasing decision-making speed. No off-line, a-priori, ordering of agents to determine the order in which they should perform their computations is necessary. Furthermore, under the given assumptions, the approach ensures that the solutions of the individual agents converge to local actions that are globally optimal up to a user-defined accuracy.

E. Comparison with distributed constraint optimization

At a first glance, the approach we propose may seem similar to approaches from the field of distributed constraint optimization (DCOP), e.g., the recently proposed ADOPT algorithm [10]. Indeed, as we will see, our approach relies on forming a tree-shaped communication structure between agents and passing of desired values for variables from parents to children, and information about optimality from children to parents, as also is the case in ADOPT. However, our approach considers a significantly different problem class than techniques used in the field of DCOP. In particular:

- DCOP addresses distributed solution of problems involving discrete variables and constraints between these, whereas our approach addresses problems involving continuous variables and constraints between these.
- DCOP approaches are typically based on ideas from the field of integer and discrete programming, e.g., branch-and-bound methods. The approach we propose is based on Lagrange theory, is developed for continuous programming problems, and includes well-established results for convergence to optimal solutions.
- DCOP approaches consider constraints between discrete variables, of which the domain of possible values of a particular variable is independent of the values of other variables. In our approach, the values that the variables controlled by the agents can take on are constrained by local dynamics of an agent's subnetwork and indirectly by the values of variables of neighboring agents.

F. Outline

The remainder of this paper is outlined as follows. In Section II we introduce a model for structuring large-scale control problems and decision-making schemes. In Section III we discuss the original serial approach in terms of this model and point out some of its drawbacks. In Section IV we propose the parallelized version, and in Section V we give an example illustrating and comparing the performance of the two approaches based on a simulation study.

II. Models for decision making

For n subproblems we define the set of nodes $\mathscr{N} = \{1,2,\ldots,n\}$ and the set of edges $\mathscr{E} = \{(i,j) \in \mathscr{N}^2 | i \neq j, \text{subproblems } i \text{ and } j \text{ depend on each other}\}$. We consider problems for which the overall control problem at control cycle k defined over a prediction horizon of N cycles can be written in an MPC fashion as [4]:

$$\min_{\substack{\tilde{x}_1,\tilde{x}_2,\ldots,\tilde{x}_n \\ \tilde{u}_1,\tilde{u}_2,\ldots,\tilde{u}_n}} \sum_{i=1}^{n} \tilde{J}_i(\tilde{x}_i,\tilde{u}_i) \tag{1}$$

$$\text{subject to } \tilde{g}_i(\tilde{x}_i,\tilde{u}_i,\tilde{w}_{\text{in},j_1,i},\ldots,\tilde{w}_{\text{in},j_{m,i},i}) = 0 \tag{2}$$

$$\tilde{h}_i(\tilde{x}_i,\tilde{u}_i) \leq 0 \tag{3}$$

$$\tilde{w}_{\text{in},j,i} = \tilde{w}_{\text{out},i,j} \tag{4}$$

$$\tilde{w}_{\text{out},j,i} = \tilde{C}_{j,i}[(\tilde{x}_i)^T \ (\tilde{u}_i)^T]^T \tag{5}$$

for $j \in \mathscr{N}, i \in \mathscr{N}, (j,i) \in \mathscr{E}$, $j_1,\ldots,j_{m_i}$ are the indices of the elements of $\{j|(j,i) \in \mathscr{E}\}$, and where for subnetwork i, $\tilde{J}_i$ is the local objective function, $\tilde{x}_i = [(x_{i,k+1})^T,\ldots,(x_{i,k+N})^T]^T$ are the subnetwork states, $\tilde{u}_i = [(u_{i,k})^T,\ldots,(u_{i,k+N-1})^T]^T$ are local inputs, and $\tilde{g}_i = [g_{i,k},\ldots,g_{i,k+N-1}]^T$ and $\tilde{h}_i = [h_{i,k},\ldots,h_{i,k+N-1}]^T$ are local equality and inequality constraints, respectively. In a similar way, we define variables $\tilde{w}_{\text{in},j,i}$ as interconnecting inputs and $\tilde{w}_{\text{out},j,i}$ as interconnecting outputs. These variables are used to define the interconnecting constraints (4) between subnetworks i and j. Matrix $\tilde{C}_{j,i}$ is an interconnecting output selection matrix that selects which local variables of subnetwork i are interconnecting outputs with respect to subnetwork j. The equality constraints (2) include the predictions of the subnetwork dynamics, e.g., equations of the form $x_{k+1,i} = f_i(x_{i,k},u_{i,k},w_{\text{in},i,k})$, where f_i is the prediction model for subnetwork i, while $\tilde{h}_i$ mainly contains domain constraints on the local states and inputs. Note furthermore that the overall objective function defined in (1) consists of the combination of the local objective functions of each agent. So, each agent has only local goals, like minimizing local flows and inputs.

A. Problem topology

To make the structure of an overall control problem more clear and see how parallelization can be used, we introduce the concept of a *problem topology*. Given the decomposition of the overall control problem into subproblems (e.g., based on geographical areas), a problem topology is the unique undirected graph representing the dependencies of subproblems on one another. Each node represents a subproblem, while an edge between two nodes indicates that the two subproblems represented by the nodes depend on each other. Since any subproblem depends on itself, self-dependence edges are not considered. For a given decomposition of the overall problem, the associated problem topology is simply found by placing edges between any two nodes representing subproblems that depend on each other.

There are different types of problem topologies, differing in additional assumptions made on the set of edges $\mathscr{E}$. Throughout the paper, we assume:

Assumption 2.1: The problem topology under consideration is a *tree topology*, i.e., a connected topologies without any cycles.

Although this assumption is somewhat restrictive, before being able to determine how to parallelize general topologies, we first have to understand how to do this for tree topologies. Once this is understood, the approach used may be extended to deal with cycles and therefore general topologies. Also, in practice it may be possible to construct a tree topology from a general topology by grouping the subproblems causing the non-tree structure, i.e., cycles, into one subproblem.

B. Decision-making schemes

As mentioned in Section I, the decision-making schemes that we consider operate by performing at each control cycle a number of iterations. The iterations terminate when a stopping criterion is satisfied, after which actions are implemented and the next cycle is started. We consider as overall stopping condition

$$\|v\|_\infty \leq \varepsilon, \tag{6}$$

where ε is a small positive number, $v \in \mathbb{R}^m, m = \sum_{i=1}^n m_i$ characterizes the interconnecting constraints of all subnetworks, i.e., $v = [\tilde{w}_{\text{in},j_1,1} - \tilde{w}_{\text{out},1,j_1}, \ldots, \tilde{w}_{\text{in},j_{m_i},1} - \tilde{w}_{\text{out},1,j_{m_1}}, \ldots, \tilde{w}_{\text{in},j_1,n} - \tilde{w}_{\text{out},n,j_1}, \ldots, \tilde{w}_{\text{in},j_{m_i},n} - \tilde{w}_{\text{out},i,j_{m_n}}]^T$, and $\|\cdot\|_\infty = \max_i |v_i|$ denotes the infinity norm, where v_i is the ith element of v. The stopping criterion is thus an upper bound condition on the difference between values that different agents want to assign to interconnecting variables, e.g., on how much flow should go from one subnetwork into another. The condition is more accurate with ε approaching zero. By varying ε a trade-off is made between the accuracy of the solution and the number of iterations required before termination.

Each iteration can be split into two phases:

- **Phase 1** is an optimization phase in which the agents solve their local subproblems.
- **Phase 2** is a stopping detection phase in which the agents determine whether the iterations should stop.

When all agents have determined that they should stop, the agents implement their actions. The agents use a set of attributes to store information, and tokens and flags to determine what to do.

1) Attributes: Agent i solving the subproblem of node i has access to the following attributes of node i:

- The *neighbors* attribute $\mathscr{N}_i$ is the set of nodes to which node i has an edge, i.e., $\mathscr{N}_i = \{j | (i,j) \in \mathscr{E}\}$. This set is initialized at the beginning of the first control cycle and stays fixed over further control cycles. A grouping of the neighbors is made using two attributes:
  The *parent* attribute P_i refers to the node $j \in \mathscr{N}_i$ that had its subproblem solved right before the node i's subproblem was considered. The *children* attribute $\mathscr{C}_i$ is the set of all nodes except the parent node, i.e., $\mathscr{C}_i = \mathscr{N}_i \setminus \{\text{P}_i\}$. The parent and children attribute are set when an agent performs its first computation in the first iteration of the first control cycle, after which they stay constant over all further iterations and cycles.
- The *local optimality* attribute LO_i indicates whether or not the agent of node i has made its decision on the local variables and interconnecting variables. This attribute is updated at the end of Phase 2 of each iteration. The local stopping criterion for agent i is given by $\max \|v^{(i)}\|_\infty \leq \varepsilon$, where $v^{(i)}$ is a vector with the evaluations of the interconnecting constraints in which variables of agent i are involved.
- The *subgroup optimality* attribute SGO_i indicates whether or not the agent of node i has local optimality and all its children have the subgroup optimality attribute positively set, i.e., $\text{SGO}_i = \text{LO}_i \wedge \left(\bigwedge_{j \in \mathscr{C}_i} \text{SGO}_j\right)$, with $\bigwedge_{j \in \emptyset} \text{SGO}_j = \text{true}$. This attribute is updated after the local optimality attribute has been updated.

2) Tokens and flags: To indicate which agents are solving their subproblems, we introduce the concept of a *computation token*. The computation token allows the agent that has a token to perform computations related to solving its subproblem, i.e., Phase 1.

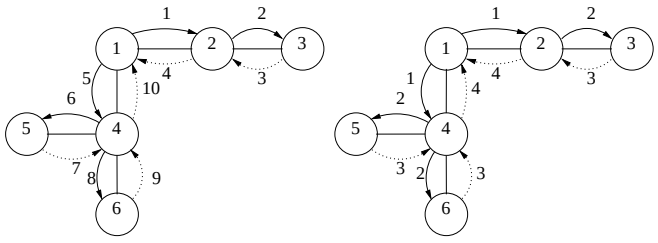
To determine whether an agent should stop, i.e., whether Phase 2 can start, an agent waits until it has received all relevant information from the agents that it requires information from. The *stop-determination flag* indicates whether an agent has all necessary information.

3) Local optimality determination: When the stop-determination flag is positively set for an agent, the agent has to determine whether or not its local solution satisfies the stopping condition. For this to be possible, we have the following.

Lemma 2.2: The agents can in a distributed way determine whether the overall stopping condition is satisfied using local stopping conditions.

Proof: The infinity norm involved in the overall stopping condition (6) can be written as

$$\begin{aligned}\|v\|_\infty &= \max_i |v_i| = \max(|v_1|, \ldots, |v_{m_1}|, \ldots, |v_{n-m_n}|, \ldots, |v_n|) \\ &= \max\left(\max(|v_1|, \ldots, |v_{m_1}|), \ldots, \max(|v_{n-m_n}|, \ldots, |v_n|)\right)\end{aligned}$$



(a) Serial (b) Parallelized

Fig. 1: Example of the order in which tokens can go. Solid arrows indicate computation tokens; dotted arrows indicate subgroup optimality information. The edges are labeled with the step within the iteration at which the information is sent.

$$= \max(\|v^{(1)}\|_\infty, \ldots, \|v^{(n)}\|_\infty).$$

where $v^{(i)}$ are the variables of subnetwork i, e.g., $v^{(1)} = [v_1, \ldots, v_{m_1}]^T$. Thus, the overall stopping condition is satisfied when

$$\max(\|v^{(1)}\|_\infty, \ldots, \|v^{(n)}\|_\infty) \leq \varepsilon, \tag{7}$$

which is true if and only if $\left(\|v^{(1)}\|_\infty \leq \varepsilon\right) \wedge \ldots \wedge \left(\|v^{(n)}\|_\infty \leq \varepsilon\right)$. Local optimality LO_i for subnetwork i is concluded when the local stopping criterion $\|v^{(i)}\|_\infty \leq \varepsilon$ is satisfied. If all agents have concluded local optimality, then $\|v^{(i)}\|_\infty \leq \varepsilon$ for each subnetwork and therefore (7) holds, and thus the overall stopping criterion (6) holds. ■

4) Global optimality determination: To determine when all agents have solved their subproblems and the agents can implement the determined actions, we have the following:

Proposition 2.3: For a tree topology of an overall convex control problem, if for a node $i \in \mathcal{N}$ each of its neighbors $j \in \mathcal{N}_i$ has the subgroup optimality flag positively set, i.e., $\mathrm{SGO}_j = \text{true}$, and if its local optimality flag is set, i.e., $\mathrm{LO}_i = \text{true}$, then the solution of the overall problem has been reached within the specified accuracy.

Proof: Since all neighbors of node i have the subgroup optimality flag positively set, the children of these neighbors and children of children, and so on, also have the subgroup optimality flag positively set. Since the subgroup optimality flag of a node can only be positively set if the node has local optimality, all children and children of children, etc. have solved their local subproblems. Thus, together with local optimality of node i, all nodes will have local optimality. Furthermore, due to the convexity of the overall control problem, the overall solution has been reached. ■

The optimization problem defined by (1)–(5) is convex, when the functions J^i and h^i are convex and the functions g^i are affine. A typical situation like this occurs when quadratic local objective functions are taken (e.g., obtained as second-order approximation of a nonlinear objective function) with linear prediction models for the subnetwork dynamics (e.g., obtained as linearization of a nonlinear model of the dynamics), defined over variables that take on their values from closed convex sets of real numbers.

III. Original serial approach

In the original serial approach, i.e., the approach of [9], one agent at a time performs computations. Thus, per iteration there is exactly one computation token. The following example illustrates the workings of the scheme.

Example 3.1 Consider the problem topology in Figure 1a. Agent i has to solve the subproblem of node i. Agent 1 starts the iterations by receiving the computation token. To determine subgroup optimality, it solves its subproblem, sends the determined desired values for the interconnecting variables to its neighbors, i.e., agents 2 and 4, and gives the computation token to one of its children from which it has not received subgroup optimality information in this iteration yet, e.g., agent 2. Agent 2 receives the token. It solves its subproblem, sends the information found to its neighbors, 1 and 3, and sends the computation token to 3 from which it has not received subgroup optimality in this iteration yet.

Agent 3 receives the token, solves its local problem, sends the information found to its neighbors. Since it has no child from which it has not received the subgroup optimality information yet, it has all up-to-date information from its neighbors, plus its own up-to-date information and therefore it can evaluate its local stopping criterion. Then, it determines its subgroup optimality and sends the subgroup optimality information to its parent, agent 2.

Agent 2 has no other child from which it has not received subgroup optimality information. The stop-determination flag for node 2 is thus true and agent 2 subsequently has to evaluate the local stopping criterion and determine subgroup optimality. It passes the subgroup optimality information to its parent, agent 1.

Agent 1 has not yet received the subgroup optimality information from agent 4, so it sends the computation token to 4. Agent 4 receives the token and takes actions to obtain the required information from its children. Ultimately, 1 receives from 4 the subgroup optimality information. Agent 1 then has received updated subgroup optimality information from all its children and evaluates its own local stopping criterion and subgroup optimality.

The iterations continue until all agents have the local stopping criterion satisfied. Using Proposition 2.3 agent 1 determines whether a next iteration has to be started, or whether the agents can implement their determined actions.◇

The serial scheme just illustrated has some drawbacks:

- only one agent is computing at a time, making iterations take a long time when there are many agents;
- even when an agent has local optimality, it will keep on performing its local optimization, even though its solution already satisfies the stopping condition, therewith increasing running time;
- iterations are always done over the whole group of agents, even though parts of the group may already have reached local or even subgroup optimality.

In the next section we propose an extension of the original scheme that addresses these drawbacks.

IV. Parallelization of the serial scheme

We propose an extension of the serial approach based on parallelization. With parallelization instead of having one agent at a time solving its subproblem, there are multiple agents at the same time working on different subproblems. Instead of having one group of agents over which the serial scheme iterates, there are several groups in which the serial scheme iterates in parallel.

Problems can be solved in parallel when they are independent of each other. By Assumption 2.1 the problem topology is connected, which means that indirectly all subproblems in the problem topology depend on each other. However, while the agents are performing their iterations to find a solution to the overall problem, the subproblems do become independent as information from locally solved problems becomes available, since within an iteration agents determine the values of their local variables once, after which they keep these values fixed throughout the current iteration. Moreover, after an agent decides on local optimality, it will keep its variables fixed, also over future iterations of the current cycle. Thus, the independency holds either only within the current iteration or also over all future iterations of the current cycle. We have:

Proposition 4.1: For a tree topology, after an agent has solved its local subproblem, its children can solve their subproblems in parallel within the current iteration.

Proof: When agent i has solved its local subproblem, the values it has determined for its variables, including the interconnecting variables, are fixed for the current iteration. Thus given these fixed values the subproblem of each child $j \in \mathscr{C}_i$ will be independent of the subproblem of agent i. Furthermore, due to the tree topology assumption, all subproblems of the descendants of child j are independent of the descendants of each other child $k \in \mathscr{C}_i \setminus \{j\}$. Therefore, the children of agent i can solve their problems in parallel. However, the group of agents representing the subproblems in the branches leaving the current node cannot be separated completely, since at the next iteration the values of the current agent may change again. ■

Proposition 4.2: For a tree topology, if a node has the local optimality flag set positively, then the branches leaving from this node can be solved in parallel within the current iteration *and* within all future iterations of the current cycle.

Proof: By Lemma 2.2, for a node $i \in \mathscr{N}$ that is locally optimal the values of its variables, including those of interconnecting variables, satisfy the stopping condition. Furthermore, although the values of the variables may change due to arrival of new information, the local stopping criterion will still be met. Due to the tree topology assumption, the branches leaving from node i are not connected to each other and therefore represent independent subproblems (given the fixed variables of the node i). Therefore each of the subproblems of the children of node i can be solved in parallel, in the current iteration and for future iterations of the current cycle. So, the group of agents solving the subproblems in the branches of node i can be grouped, and within this group the serial scheme can be performed. ■

Example 3.1 revisited We reconsider Example 3.1, now using the parallelized serial approach. Figure 1b shows the schematics of the order in which agents work. Agent 1 starts by receiving the computation token. It solves its subproblem and sends the results of this to agents 2 and 4. To determine subgroup optimality agent 1 has to receive subgroup optimality from these agents. By Proposition 4.1 it sends a computation token to each of its two children. Thus, 2 and 4 each receive a computation token. They solve their local problems and send the obtained information to their neighbors, i.e., agent 3, and agents 3 and 5, respectively. To determine subgroup optimality they have to obtain subgroup optimality from their children. Agent 2 has no children. Therefore, agent 2 determines subgroup optimality and returns this information to 1. However, agent 4 has children, so by Proposition 4.1 it sends computation tokens to these.

In the meantime, agent 1 has received the subgroup optimality information of 2. However, since 1 has not received this information of 4 yet, its stop-determination flag is still false. It cannot yet proceed to determine on its own subgroup optimality and decide whether or not to start a new iteration.

When agent 4 has received the subgroup optimality information of 5 and 6, it determines its own subgroup optimality and sends the result to its parent, 1. Agent 1 has then a positive stop-determination flag; thus, it decides on whether or not to start a new iteration. Since no agent has concluded local optimality, 1 starts a new iteration.

Suppose that after some iterations agent 4 reaches local optimality. The values of its interconnecting variables will stay fixed over the following iterations. It notifies this to all its neighbors, therewith indicating that these neighbors should also not update their interconnecting variables with respect to agent 4 anymore. The only task remaining for 4 is to inform its parent of subgroup optimality, such that at some point the stop-determination flag of its parent will be true, therewith allowing its parent to also determine subgroup optimality. By Proposition 4.2 the children of 4 can solve their problems in parallel over all future iterations. As long as 4 does not receive positive subgroup optimality flags from its children, it will not send anything to its parent, 1.

In the meantime, when the parent of agent 4 has received the subgroup optimality flags of the children that have not yet indicated local optimality, i.e., 2, the parent assumes negative subgroup optimality for the children that do have local optimality, but that do not have not reported positive subgroup optimality yet.

Each of the agents solving the subproblems of the children of 4 will get similar roles as agent 1. They know that their parent, 4, has local optimality, and that it will therefore not change the values of its interconnecting variables and not send them further updates. The agents of the children of 4 continue solving the subproblems of their branches and report to 4 when they have reached subgroup optimality. When agent 4 receives this information, it sends this to its parent, ultimately leading to stopping of the iterations. ◇

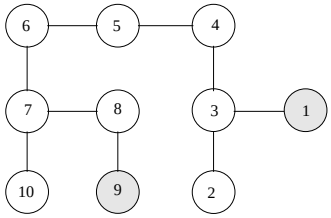


Fig. 2: Problem topology for 10 subnetwork problem with disturbances in subnetwork 1 and 9.

agent	1	2	3	4	5	6	7	8	9	10
serial	23	23	23	23	23	23	23	23	23	23
parallelized	21	22	25	25	1	1	20	18	18	10

TABLE I: Number of steps per agent. In total 230 steps are performed using the serial approach, whereas 161 steps are performed using the parallelized serial approach.

central	19.00
serial	19.01
parallelized	19.05

TABLE II: Costs of the control (per unit).

V. Comparison

We illustrate the performance of the schemes using the problem topology depicted in Figure 2, representing a load-frequency control problem from the domain of power network control [11], [12]. Load-frequency control involves keeping power consumption and generation equal. In this study, the agents of 10 subnetworks control the adjustment of generation after a load change in subnetwork 2 and 9. Since the subnetworks are connected to each other, in order to predict the evolution of their local subnetwork, the agents have to agree with each other on the flows of power between the subnetworks. Agent 5 initiates the first iteration. More details on the models used can be found in [9]. The overall problem satisfies the assumptions made in previous sections.

Table I shows the number of steps before the agents finish their computations for the serial and parallelized schemes when $\varepsilon = 0.001$ is taken. For the serial scheme 23 steps are required for each agent, yielding in total 230 computation steps performed serially. In the parallelized scheme agents 5 and 4 already after one step determine local optimality. Thus, the subproblems of their neighbors are solved in parallel, speeding up the total decision making time. Table II shows the costs of the actions determined by each scheme and the costs of actions that a centralized agent would determine, i.e., the ideal case. The performance of the serial scheme is almost as good as the centralized control. The parallelized scheme has slightly higher costs than the serial scheme, since in the parallelized scheme an individual agent stops updating its variables at the moment that its local stopping criterion is satisfied, whereas in the serial approach an agent will also after this keep updating its variables, until all agents stop.

VI. Conclusions & future research

In this paper we have considered a scheme for multi-agent control of, e.g., large-scale networks. We have introduced problem topologies and decision-making schemes, explained how a recently introduced scheme based on iterations of serial computations by multiple agents fits into this point of view, and pointed out some flaws in the serial scheme that make decision making slow down when applied to large-scale networks. For tree-structured problem topologies with convex overall problems as solution to this we have proposed parallelization of the serial scheme. We have illustrated our approach with an example, that showed the speed up of the parallelized approach in a simulation study.

Topics for future research are extending the approach to deal with general problem topologies and comparing the resulting approach with parallel approaches based on the auxiliary problem principle. Moreover, we will consider an approach in which initially each agent operates solely by itself and will involve other agents only when it finds this necessary, contrarily to first involving all agents and then reducing to smaller groups. Our future research will also consider a hybrid approach for controlling systems with both continuous and discrete elements. This approach will combine the current scheme for dealing with continuous variables with a scheme from the field of distributed constraint optimization to deal with discrete variables [10].

Acknowledgments

Research supported by the project “Multi-agent control of large-scale hybrid systems” (DWV.6188) of the Dutch Technology Foundation STW, the European 6th Framework Network of Excellence “HYCON” (FP6-IST-511368), BSIK project “Next Generation Infrastructures (NGI)”, an NWO Van Gogh grant (VGP79-99), and the Transport Research Centre Delft.

References

[1] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. USA: MIT Press, 2000.

[2] K. P. Sycara, “Multiagent systems,” *AI Magazine*, vol. 2, no. 19, pp. 79–92, 1998.

[3] D. D. Siljak, *Decentralized Control of Complex Systems*, ser. Mathematics in Science and Engineering. Boston, Massachusetts: Academic Press, Inc., Jan. 1991, vol. 184.

[4] J. M. Maciejowski, *Predictive Control with Constraints*. Harlow, England: Prentice Hall, 2002.

[5] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, “Distributed model predictive control,” *IEEE Control Systems Magazine*, vol. 1, pp. 44–52, Feb. 2002.

[6] P. Hines, L. Huaiwei, D. Jia, and S. Talukdar, “Autonomous agents and cooperation for the control of cascading failures in electric grids,” in *Proceedings of the 2005 IEEE International Conference on Networking, Sensing and Control*, Tucson, Arizona, Mar. 2005, pp. 273–278.

[7] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Nashua, New Hampshire: Athena Scientific, 1997.

[8] C. B. Royo, “Generalized unit commitment by the radar multiplier method,” Ph.D. dissertation, Technical University of Catalonia, Barcelona, Spain, May 2001.

[9] R. R. Negenborn, B. De Schutter, and J. Hellendoorn, “Multi-agent model predictive control for transportation networks: Serial versus parallel schemes,” in *Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2006)*, Saint-Etienne, France, May 2006, pp. 339–344.

[10] P. J. Modi, W. M. Shen, M. Tambe, and M. Yokoo, “ADOPT: Asynchronous Distributed Constraint Optimization with quality guarantees,” *Artificial Intelligence*, vol. 161, no. 1-2, pp. 149–180, Jan. 2005.

[11] P. Kundur, *Power System Stability and Control*. New York: McGraw Hill, 1994.

[12] P. W. Sauer and M. A. Pai, *Power System Dynamics and Stability*. London, UK: Prentice-Hall, 1998.