2013 American Control Conference (ACC)
Washington, DC, USA, June 17-19, 2013

Distributed Parameterized Model Predictive Control of Networked Multi-Agent Systems

Greg Droge, Magnus Egerstedt

Abstract—**When considering a distributed control framework for a multi-agent system, care has to be taken to respect the limited information available to each agent as well as the amount of possible communication in the network. This presents difficulties in performing distributed model predictive control as an agent must typically be able to simulate its neighbors dynamics into the future. We present a framework based on parameterized feedback control laws which will allow a multi-agent system to perform distributed model predictive control. It enables both the simulation of neighbors' states as well as the ability to minimize a collective cost in a distributed fashion. Moreover, given cost and dynamic dependencies between agents, we characterize the information that is needed by each agent to evaluate whether the optimization is feasible in a given network.**

I. Introduction

Model predictive control (MPC) is a control scheme which draws upon the key benefits of optimal control while adding an element of feedback by repeatedly calculating the optimal control solution during execution, e.g., [1], [2]. This is desirable as optimal control is a technique which uses a dynamic model of the system to simulate the state trajectory into the future and choose an optimal control action according to some defined cost, e.g. [3]. However, due to the nature of optimal control typically being open loop, uncertainties diminish the validity of the solution which is why MPC is often used as a feedback mechanism, e.g., [1].

In this paper, we consider a framework based on parameterized feedback control laws for performing model predictive control of multi-agent systems in a distributed manner. In other words, we introduce a framework to allow a group of agents to work together to minimize a cost without any central computing component or any one agent performing a large portion of the computation. A key element which makes this different from distributed parameter optimization, e.g. [4], [5], [6], is that a typical cost not only depends on the current state, but rather a continuum of future states. This adds two elements of difficulty when considered in the multi-agent scenario. First, each agent must be able to either communicate or simulate neighboring agents actions. Second, a way must be formulated in which agents can influence the actions of neighboring agents in order to come to a collective minimum.

Email: {gregdroge,magnus}@gatech.edu.
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA.

These issues have been addressed in one manner or another in a number of different multi-agent MPC frameworks, e.g. [7], [8], [9], [10]. To avoid the possibly burdensome task of communicating trajectories, authors in [9] introduce constraints to the problem which alleviates any need for communicating trajectories. To overcome difficulties associated with the distributed optimization of trajectories, [7] and [9] provide methods which avoid the collaborative optimization, but have convergence guarantees. In another method, [8] devises a framework to distributively optimize over the discretized state trajectory of linear systems. Finally, [10] presents a method of distributed optimal control, but it is limited to linear systems which exhibit a hierarchy of knowledge and decision making.

Throughout the remainder of this paper we will present an MPC framework which will allow a multi-agent system to overcome the mentioned difficulties. In the next section we formulate preliminaries for multi-agent parameterized MPC and discuss how it allows agents to overcome communication difficulties. Section III will then address how the formulation allows the distributed optimization technique of dual-decomposition, e.g. [4], to be used by agents to negotiate future trajectories. In section IV, information requirements will be discussed to determine if a given network of communication is sufficient to perform distributed parameterized MPC. The paper will end with an example in Section V and some concluding remarks in Section VI.

II. Preliminaries

This section will introduce the basic concept of parameterized, multi-agent MPC. This will be done by first introducing the problem solved by the multi-agent parameterized MPC formulation. The ability for this method to overcome communication difficulties will then be addressed. Finally, notation used throughout the remainder of the paper will be introduced.

A. Problem Definition

Model predictive control is a method of adding feedback to the otherwise typically open loop optimal control solution, e.g. [1]. The basic idea is to use optimal control techniques to calculate an optimal control trajectory over some pre-defined time horizon, apply the first input, and repeat at the next time step.

While MPC is able to utilize the benefits of optimal control, one potential drawback is the cost of computing the optimal control solution at every time instant. In [11], the concept of utilizing parameterized feedback control laws as a means to alleviate this computational burden was introduced, effectively exchanging the two-point boundary value problem for a parameter optimization problem. While parameterized MPC has computational benefits in the single agent scenario, it offers even more benefits in the multi-agent scenario by providing a paradigm on which to communicate and negotiate over future trajectories.

To explicitly represent the multi-agent scenario, assume that each agent's dynamics are of the form $\dot{x}_i(t) = f_i(x_i(t), u_i(t))$, where $x_i(t)$ is the state and $u_i(t)$ is the control input of agent i at time t. Also, assume that each agent will execute a feedback law of the form $\kappa_i(x_i(t), x^d_{-i}(t), \theta_i)$ where $x^d_{-i}(t)$ is a set of the states upon which agent i's dynamics depend and θ_i is a vector of tunable parameters. Without loss of generality, this will allow the dynamics to be expressed as:

$$\dot{x}_i(t) = f_i(x_i(t), x^d_{-i}(t), \theta_i). \tag{1}$$

In order to chose θ_i at each time instant, we consider the following general form for the collective cost to be minimized at each time step:

$$J = \sum_{i=1}^{N} J_i, \tag{2}$$

where J_i is the cost assigned to agent i and takes the form

$$J_i = \int_{t_0}^{t_f} L_i(x_i(t), x^c_{-i}(t), \theta^c_{-i})dt + \phi_i(x_i(t_f), x^c_{-i}(t_f), \theta^c_{-i}), \tag{3}$$

subject to (1), where x^c_{-i} represents the set of agents that agent i is coupled to through its cost, θ^c_{-i} is the parameter vectors corresponding to those agents, t_0 is the initial time, $t_f = t_0 + \Delta$, and Δ is the time horizon being evaluated.

The division of the cost is left to the designer of the system. A method proposed in [12] recommends separating the cost into components which are not additively separable. However, the authors of [9] mention that adding extra terms in the cost can improve the results. For example, if the non-separable J_i was dependent on both x_j and x_k, then it may be good for convergence to also assign agent i any terms in J which deal exclusively with x_j and x_k. This will allow agent i to better model agents j and k.

B. Communication of Trajectories

A key aspect of this formulation is the ability for agents to communicate entire trajectories by solely communicating a number of parameters. This is possible due to the fact that the trajectory of each agent will be determined by its parameter vector, initial conditions, and the feedback control law it is executing. Therefore, to communicate a trajectory to a neighbor, an agent need only communicate these three pieces of information. The neighboring agent will then be able to calculate the trajectory of its neighbor by simulating it forward in time. Thus, there is an inherent tradeoff between communication and computation.

C. Notation

With the problem to be solved in mind, we outline the notation to be used throughout the paper. First note that, as discussed in more detail in Section III, dual-decomposition requires each agent to have a local version of all the variables associated with the agents to which it is coupled. As such, we use the subscripts a_{ij} to denote agent i's version of agent j's variable a.

We now define the following:

- $\mathcal{N}^d_i \equiv \{j : \frac{\partial f_i}{\partial x_j} \neq 0\}$: The agents to which agent i is dynamically coupled.
- $\mathcal{N}^d_{-i} \equiv \{j : \frac{\partial f_j}{\partial x_i} \neq 0\}$: The agents that are dynamically coupled to agent i.
- $\mathcal{N}^c_i \equiv \{j : \frac{\partial L_i}{\partial x_j} \neq 0$ or $\frac{\partial \phi_i}{\partial x_j} \neq 0\}$: The agents to which agent i is coupled to through cost.
- $\mathcal{N}^I_i$: The set of agents from which agent i will require information.
- $x_{ij}(t; t_0)$: State of agent j at time t as predicted by agent i at time t_0.
- $x^d_{i,-j}(t; t_0) \equiv \{x_{ik}(t; t_0) : k \in \mathcal{N}^d_j\}$: Agent i's versions of the states agent j depends on through agent j's dynamics.
- $x^c_{i,-j}(t; t_0) \equiv \{x_{ik}(t; t_0) : k \in \mathcal{N}^c_j\}$: Agent i's versions of the states agent j depends on through agent j's costs.
- θ_{ij}: Agent i's version of agent j's parameter vector.
- f_j: The dynamics used to compute x_{ij}.

We also use the following abbreviations to simplify our expressions:

- x_{ij} or $x_{ij}(t)$: We use this to refer to $x_{ij}(t; t_0)$

- $x^d_{i,-j}$ or $x^c_{i,-j}$: We use this to refer to $x^d_{i,-j}(t;t_0)$ or $x^c_{i,-j}(t;t_0)$ respectively.
- f_{ij}: We use this to refer to

$$f_j(x_{ij}(t,t_0), x^d_{i,-j}(t;t_0), \theta_{ij})$$

- $\bar{\theta}_i \equiv \{\theta_{ij} : j = 1, ..., N\}$: All parameter vectors that agent i could depend on.

Furthermore, we can represent cost and dynamic dependencies through directed graph structures which will induce an undirected information dependency graph. We use $\mathcal{G}^c(V, E^c)$ to denote a graph where the node v_i corresponds to agent x_i and a directed edge $(v_j, v_i) \in E^c$ iff $j \in \mathcal{N}^c_i$. Similarly we use $\mathcal{G}^d(V, E^d)$ to denote the dynamic dependency graph where $(v_j, v_i) \in E^d$ iff $j \in \mathcal{N}^d_i$. In Section IV we will then give conditions on $\mathcal{G}^c$ and $\mathcal{G}^d$ which will induce an undirected information graph $\mathcal{G}^I(V, E^I)$ where an edge $(v_i, v_j) \in E^I$ exists iff it is necessary for agents i and j to exchange information, thus building the set $\mathcal{N}^I_i$. An example of these three graphs is shown in Figures 2.

III. Multi-agent Distributed Parameterized MPC

The previous section presented problem formulation for multi-agent parameterized MPC and addressed how using parameterized feedback laws can facilitate the communication of trajectories between agents. In this section, it is shown that the use of these parameterized feedback laws also facilitates the negotiation between agents, allowing them to arrive at a collective minima. This is done by first showing that the distributed parameter optimization technique of dual decomposition can be used for the distributed optimization of (2). This will lead to the need for gradients of the cost, which are then formulated. Finally, the section ends with a summary of the distributed parameterized MPC algorithm.

A. Dual-Decomposition

Dual-decomposition is a tool which allows for a parameter optimization problem to be solved in a distributed fashion, e.g., [4], [6], [8], [13], [14]. The formulation considered in this paper is very similar to that found in [13] as well as [14].

In order for (2) to be decomposed into subproblems that each agent can work on, allow each agent to maintain its own "version" of it's neighbors' variables with the constraint that each agent's "version" of the variables be equal. Namely, let agent i maintain x_{ij}, θ_{ij} $\forall j \in \mathcal{N}^I_i$ and rewrite (2) as:

$$J = \sum_{i=1}^{N} J_i. \tag{4}$$

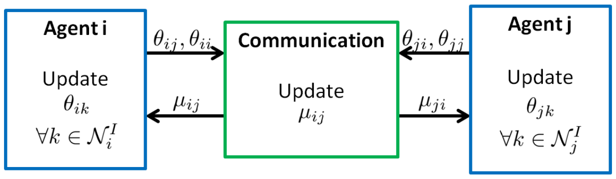
Agent i
Update
θ_{ik}
$\forall k \in \mathcal{N}^I_i$
θ_{ij}, θ_{ii}
μ_{ij}
Communication
Update
μ_{ij}
θ_{ji}, θ_{jj}
μ_{ji}
Agent j
Update
θ_{jk}
$\forall k \in \mathcal{N}^I_j$

Fig. 1. This shows how agents would use equations (6) through (9) to perform the dual decomposition.

$$\text{s.t. } \theta_{ij} = \theta_{jj}, i = 1, ..., N j \in \mathcal{N}^I_i$$

Note that if $x_{ij}(t_0) = x_{jj}(t_0)$ and x_{ij} executes the same dynamics as x_{jj} then the constraint $\theta_{ij} = \theta_{jj}$ will ensure equality of $x_{ij}(t)$ and $x_{jj}(t)$ by uniqueness of solutions to differential equations (assuming Lipshitz conditions hold, e.g., [15]).

To form the dual, Lagrange multipliers are introduced and the constraint is then moved into the cost function (e.g. [16], [17]), as follows:

$$\max_{\mu} \min_{\theta} \hat{J} = \sum_{i=1}^{N} J_i + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}^I_i} {\mu_{ij}}^T (\theta_{ij} - \theta_{ii}), \tag{5}$$

where the Lagrange multiplier vector μ_{ij} ensures equality between θ_{ij} and θ_{jj}.

In order for this problem to be solved in a distributed fashion, we use Uzawa's saddle point algorithm [18]. Uzawa's algorithm is a strategy performs gradient accent on the variables being maximized and gradient decent on those being minimized. Thus, the optimization variables can take on the following dynamics:

$$\dot{\theta}_{ij} = -\eta \frac{\partial \hat{J}}{\partial \theta_{ij}}^T \tag{6}$$

$$\dot{\mu}_{ij} = \eta \frac{\partial \hat{J}}{\partial \mu_{ij}}^T, \tag{7}$$

where the gradients in (6) and (7) can be written as:

$$\frac{\partial \hat{J}}{\partial \theta_{ij}} = \begin{cases} \frac{\partial J_i}{\partial \theta_{ij}} + \mu_{ij} & i \neq j \\ \frac{\partial J_i}{\partial \theta_{ii}} - \sum_{n \in \mathcal{N}^c_{-i}} \mu_{ni} & i = j \end{cases} \tag{8}$$

$$\frac{\partial \hat{J}}{\partial \mu_{ij}} = \theta_{ij} - \theta_{jj} \tag{9}$$

and $\frac{\partial J_i}{\partial \theta_{ij}}$ will defined in Theorem III.1.

This can be worked out as a multi-agent solution by having each agent compute the update for all of the variables it is maintaining and then communicate that update to its neighbors. Each agent also maintains a copy of the Lagrange multipliers and uses the communicated variables to calculate the updates, as is depicted in Figure 1.

Note on Optimality: One further note needs to be made on the optimization. Uzawa's algorithm is guaranteed to converge to the minimum of the original problem if the original problem is strictly convex (see for example [16], [18]). If it is not strictly convex, then the solution will converge to some local minimum with some additional conditions on local convexity and initial conditions, e.g. [17] (i.e. the problem must be well-defined for gradient strategies).

B. Gradients

As seen in (8), agents will be able to use the gradients of their individual costs to be able to collaborate with neighbors in the minimization of the collective cost. As such, we now provide a theorem giving the gradients used for optimization.

Theorem III.1. *Given a cost J_i of the form in (3), the gradient of J_i with respect to θ_{ij} is given as*

$$\frac{\partial J_i}{\partial \theta_{ij}} = \xi_{ij}^T(t_0) \tag{10}$$

where

$$\dot{\xi} = \frac{\partial L_i}{\partial \theta_{ij}}^T - \frac{\partial f_{ij}}{\partial \theta_{ij}}^T \lambda_{ij};\ \xi(t_f) = \frac{\partial \phi_i}{\partial \theta_{ij}}^T (t_f) \tag{11}$$

$$\dot{\lambda}_{ij} = -\frac{\partial L_i}{\partial x_{ij}}^T - \sum_{k \in \mathcal{N}_{-j}^d} \frac{\partial f_{ik}}{\partial x_{ij}}^T \lambda_{ik}; \tag{12}$$

$$\lambda_{ij}(t_f) = \frac{\partial \phi_i}{\partial x_{ij}}^T (t_f)$$

Proof: We first augment (3) with the dynamics and write it as

$$\hat{J}_i(\bar{\theta}_i) = \int_{t_0}^{t_f} \Big(L_i(x_{ii}(t), x_{i,-i}^c(t), \theta_{i,-i}^c) + \tag{13}$$

$$\sum_{j=1}^{N} \lambda_{ij} \big(f_{ij}(x_{ij}(t), x_{i,-j}^d(t), \theta_{i,j}) - \dot{x}_{ij} \big) \Big) dt +$$

$$\phi_i(x_{ii}(t_f), x_{i,-i}^c(t_f), \theta_{i,-i}^c)$$

Now, vary $\theta_{ij} \to \theta_{ij} + \epsilon \gamma_{ij}$ which causes the state to vary as $x_{ij} \to x_{ij} + \epsilon \eta_{ij}$. After applying traditional variational principles (e.g., [3]) and rearranging terms we can write

$$\frac{1}{\epsilon} \Big(J_i(\bar{\theta}_i + \epsilon \bar{\gamma}_i) - J_i(\bar{\theta}_i) \Big) = \tag{14}$$

$$= \sum_{j=1}^{N} \Bigg[\int_{t_0}^{t_f} \Bigg(\Big(\frac{\partial L_i}{\partial x_{ij}} + \sum_{k=1}^{N} \lambda_{ik}^T \frac{\partial f_{ik}}{\partial x_{ij}} + \dot{\lambda}_{ij}^T \Big) \eta_{ij} +$$

$$\Big(\frac{\partial L_i}{\partial \theta_{ij}} + \lambda_{ij}^T \frac{\partial f_{ij}}{\partial \theta_{ij}} \Big) \gamma_{ij} \Bigg) dt + \frac{\partial \phi_i}{\partial \theta_{ij}}(t_f) \gamma_{ij}$$

$$- \lambda_{ij}^T \eta_{ij} \big|_{t_0}^{t_f} + \frac{\partial \phi_i}{\partial x_{ij}}(t_f) \eta_{ij}(t_f) \Bigg] + o(\epsilon)$$

Note that $\frac{\partial f_{ij}}{\partial x_{ik}} = 0 \ \forall \ k \notin \mathcal{N}_j^d$ and $\eta_{ij}(t_0) = 0$. Then allow λ to be defined as (12). This permits the gradient for θ_{ij} to be expressed as

$$\frac{\partial J_i}{\partial \theta_{ij}} = \int_{t_0}^{t_f} \Big(\frac{\partial L_i}{\partial \theta_{ij}} + \lambda_{ij}^T \frac{\partial f_{ij}}{\theta_{ij}} \Big) ds + \frac{\partial \phi_i}{\partial \theta_{ij}}(t_f). \tag{15}$$

Allowing ξ_{ij} to be defined as

$$\xi_{ij}^T(t) = \int_{t}^{t_f} \Big(\frac{\partial L_i}{\partial \theta_{ij}} + \lambda_{ij}^T \frac{\partial f_{ij}}{\theta_{ij}} \Big) ds + \frac{\partial \phi_i}{\partial \theta_{ij}}(t_f). \tag{16}$$

We can differentiate (16) with respect to t to obtain the dynamics in (11) and the gradient in (10). ■

C. MPC Framework

Having the dual-decomposition framework and gradients give us two of the final pieces needed in order to outline the multi-agent parameterized MPC algorithm below. The final piece, $\mathcal{N}_i^I$, will be given in the next section.

Multi-Agent Parameterized MPC

1) Agent i communicates $x_i(t_0)$ to each agent $j \in \mathcal{N}_i^I$.
2) Agent i uses equations (6) and (8) with Theorem III.1 to update θ_{ij} $\forall j \in \mathcal{N}_i^I$.
3) Agent i communicates new values of θ_{ij} and θ_{ii} to each agent $j \in \mathcal{N}^I$.
4) Agent i uses equation (9) to update the Lagrange multipliers.
5) Each agent applies one control input
6) Repeat the entire process

IV. Induced Information Structure

While the previous sections have discussed the setup of the parameterized MPC scheme, this section will address whether or not a given network of communication will be capable of performing the distributed MPC scheme. Specifically, information requirements will be derived to determine whether or not the agents are capable of performing the algorithm with the information available to them in the network.

An information exchange between two agents will be necessary if either agent has an opinion about the other agent. This is expressed mathematically by determining whether or not each agent always has a non-zero gradient with respect to the variables associated with the other agent. Thus, the following theorem and corollary express the information required by each agent in the network.

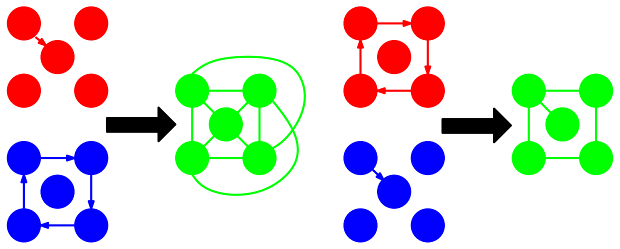

Fig. 2. This figure shows how dynamic dependencies make the required communication explode as needed information propagates through the edges in dynamic dependency graph. Cost dependence ($\mathcal{G}^c$), dynamic dependence ($\mathcal{G}^d$), and the induced information graph ($\mathcal{G}^I$) are shown in red, blue, and green respectively

Theorem IV.1. $\frac{\partial J_i}{\partial \theta_{ij}} = 0$ *for general costs and dynamics iff* $\forall\ k \in \mathcal{N}_i^c$ *there is no directed path in* $\mathcal{G}^d$ *pointing from* v_j *to* v_k.

Corollary IV.1.1. *Evaluating* $\mathcal{G}^I$ *presented in Section II-C, one result of this theorem is that* $j \in \mathcal{N}_i^I$ *iff one of the following conditions hold*

1) $\exists k \in \mathcal{N}_i^c$ *such that there is a path pointing from* v_j *to* v_k *in* $\mathcal{G}^d$
2) $\exists l \in \mathcal{N}_j^c$ *such that there is a path pointing from* v_i *to* v_l *in* $\mathcal{G}^d$

Proof: See the technical note associated with this paper [19]. ■

This shows the detrimental effect of dynamic dependence in multi-agent distributed MPC. Dynamic dependencies induce the need for information from other agents. For example, if agent i depends dynamically on agent j and agent j depends dynamically on agent k, then agent i will have an opinion about agent k. Figure 2 shows a simple case where one agent requires information from the entire system despite having no dynamic dependence itself. On the other hand, when there are no dynamic dependencies, the information graph is simply the undirected cost graph. *Therefore, when coupling between agents is done only through the costs, it permits the amount of information required to be a function of the separability of the collective cost.*

V. Example

We now present an example to illustrate the ability of the parameterized MPC framework to perform distributed MPC. There are two parts, namely:

1) *Dual*: Agents perform dual-decomposition as presented to optimize the central cost.
2) *No Dual*: Each agent performs the MPC algorithm without negotiation. At each time step each agent does the following:

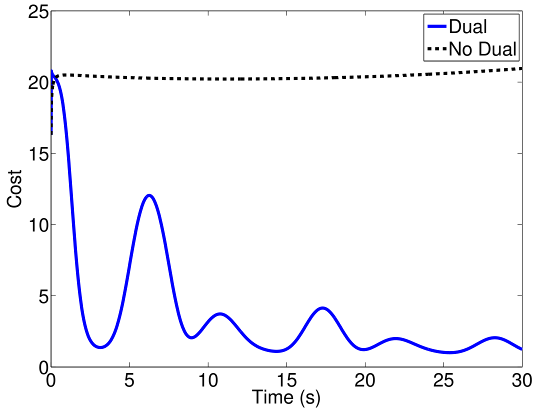

Fig. 4. This figure shows the cost versus time for the agents greedily choosing their variables (No Dual) and for the agents negotiating over the variables (Dual).

- Communicates its initial condition and parameter vector with neighbors.
- Takes a gradient step on its own parameters.

We will see that the optimization with dual-decomposition provides for much better results as the agents are able to negotiate on the parameters to be used in each mode.

The task we set out to accomplish is to have the agents spread out on a circular orbit while maintain a certain velocity. Assume that the dynamics of each agent can be represented as $\dot{x}_i = u_i$, where $x_i, u_i \in \mathbb{R}^2$. To perform a desired orbit, we parameterize the control law for orbiting given in [20] and express it as

$$\kappa_i(x_i, \theta_i) = \theta_i \begin{bmatrix} \gamma & \omega_{orb} \\ -\omega_{orb} & \gamma \end{bmatrix} \hat{x}, \tag{17}$$

where

$$\gamma = g_{lc}\Big(r^2 - \|\hat{x}\|^2\Big),$$

$\hat{x} = x - c$, $c \in \mathbb{R}^2$ is the center of the orbit, $\theta_i \in \mathbb{R}$ is a tunable gain on the speed, $g_{lc} \in \mathbb{R}$ is a gain on convergence to the limit cycle, $\omega_{orb} \in \mathbb{R}$ is the desired frequency of the orbit, and $r \in \mathbb{R}$ is the radius of the orbit. Finally, to perform orbiting, allow $u_i = \kappa_i(x_i, \theta_i)$.

To maximize distance between agents while minimizing the difference from a desired speed we present the cost

$$J_i = \int_{t_0}^{t_f} \frac{\rho_1}{2}(\theta_i - v_d)^2 dt + \tag{18}$$

$$\rho_2 \sum_{j \in \mathcal{N}^c} \sum_{k \in \mathcal{N}^c, k \neq j} \exp\Big(-g(x_j - x_k)^T(x_j - x_k)\Big)$$

where ρ_1 and ρ_2 are weights on the different costs and g is a constant. This cost will allow each agent to adjust

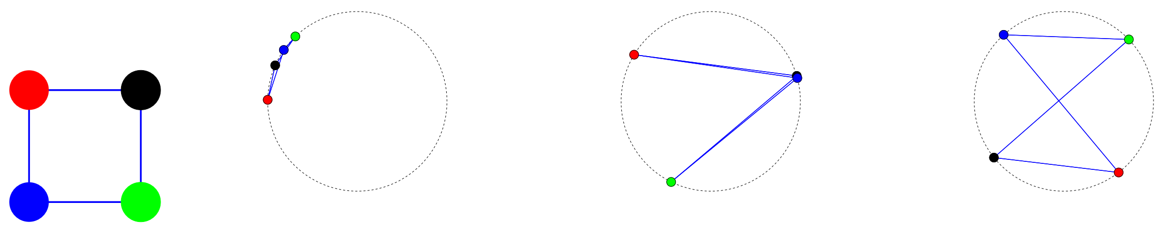

Fig. 3. This illustrates the utility of the proposed MPC framework. On the far left is shown $\mathcal{G}^I$, on the left-middle the starting configuration, right-middle the result of agents greedily choosing their optimal parameters, and on the right the result of agents negotiating to find a collective optimality point.

it's speed so that, at the end of the time window, it is as far away from its neighbors as possible.

The results of both simulations can be seen in Figures 3 and 4. When not using dual-decomposition, the agents are not able to spread out effectively. They are essentially performing a greedy optimization, doing what seems best for them without any concern for the collective whole. In Figure 3, the black and blue agents are not communicating so their greedy avoidance of the green and red agents make them come close together. However, when using the dual decomposition the agents are able to negotiate with each other to determine the best overall velocities and spread out. In Figure 4 we see that the overall cost of the system is indeed minimized and it has the typical oscillation present with dual-decomposition, e.g. [8].

VI. Conclusion

In this paper we have presented a framework based on parameterized feedback control laws for performing distributed MPC of networked multi-agent systems. This allows for the communication and optimization of state trajectories in a distributed manner. To perform distributed MPC we characterized the information necessary for each agent to perform the optimization at each step and found that dynamic dependencies cause required information between agents to propagate down dynamic dependencies whereas cost dependencies did not. We presented an example using this framework which showed that the performance can be greatly improved versus a greedy approach without negotiation.

References

[1] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[2] E. Camacho and C. Bordons, *Model predictive control*. Springer Verlag, 2004.

[3] D. Kirk, *Optimal control theory: an introduction*. Dover Pubns, 2004.

[4] D. Palomar and Y. Eldar, *Convex optimization in signal processing and communications*. Cambridge university press, 2010.

[5] D. Feijer and F. Paganini, "Stability of primal-dual gradient dynamics and applications to network optimization," *Automatica*, vol. 46, no. 12, pp. 1974–1981, 2010.

[6] J. Wang and N. Elia, "Control approach to distributed optimization," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*. IEEE, 2010, pp. 557–561.

[7] W. Dunbar and R. Murray, "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 4, pp. 549–558, 2006.

[8] P. Giselsson and A. Rantzer, "Distributed model predictive control with suboptimality and stability guarantees," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 7272–7277.

[9] T. Keviczky, F. Borrelli, and G. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, no. 12, pp. 2105–2115, 2006.

[10] P. Shah and P. Parrilo, "An optimal controller architecture for poset-causal systems," *Arxiv preprint arXiv:1111.7221*, 2011.

[11] G. Droge and M. Egerstedt, "Behavior-based switch-time mpc for mobile robots," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012.

[12] J. Shamma, *Cooperative control of distributed multi-agent systems*. Wiley Online Library, 2007.

[13] A. Rantzer, "Dynamic dual decomposition for distributed control," in *American Control Conference, 2009. ACC'09*. IEEE, 2009, pp. 884–888.

[14] H. Terelius, U. Topcu, and R. Murray, "Decentralized multi-agent optimization via dual decomposition." IFAC, 2011.

[15] H. Khalil and J. Grizzle, *Nonlinear systems*. Prentice hall, 1992, vol. 3.

[16] S. Boyd, *Convex Optimization*. Cambridge: Cambridge University Press, 2004.

[17] D. Luenberger and Y. Ye, *Linear and nonlinear programming (3rd ed.)*. Springer, 2008, vol. 116.

[18] K. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Nonlinear Programming*. Stanford University Press, Stanford, CA, 1958.

[19] G. Droge and M. Egerstedt, "Distributed parameterized model predictive control of networked multi-agent systems," School of Electrical and Computer Engineering, Georgia Institute of Technology, http://gritslab.gatech.edu/Droge/wp-content/uploads/2013/02/Memorandum_Feb_2013.pdf, Tech. Rep., February 2013.

[20] D. Kim and J. Kim, "A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer," *Robotics and Autonomous Systems*, vol. 42, no. 1, pp. 17–30, 2003.