# A general approach for consensus using optimistic planning

Lucian Busoniu, Irinel-Constantin Morarescu

# Optimistic Planning for Consensus

Lucian Buşoniu and Constantin Morărescu

*Abstract*— **An important challenge in multiagent systems is consensus, in which the agents are required to synchronize certain controlled variables of interest, often using only an incomplete and time-varying communication graph. We propose a consensus approach based on *optimistic planning* (OP), a predictive control algorithm that finds near-optimal control actions for any nonlinear dynamics and reward (cost) function. At every step, each agent uses OP to solve a local control problem with rewards that express the consensus objectives. Neighboring agents coordinate by exchanging their predicted behaviors in a predefined order. Due to its generality, OP consensus can adapt to any agent dynamics and, by changing the reward function, to a variety of consensus objectives. OP consensus is demonstrated for velocity consensus (flocking) with a time-varying communication graph, where it preserves connectivity better than a classical algorithm; and for leaderless and leader-based consensus of robotic arms, where OP easily deals with the nonlinear dynamics.**

## I. Introduction

Multi-agent systems [20] have applications in a wide variety of domains such as robotic teams, energy and telecommunication networks, collaborative decision support systems, data mining, etc. Each agent typically has only a local, limited view, which means decentralized approaches are necessary to control the overall system. In this decentralized setting, requirements on the coherent behavior of the agents are often expressed in terms of *consensus*, in which the agents must reach agreement on controlled variables of interest [15], [17], [14]. Existing approaches to consensus problems are often limited to simple, linear agents, and are each designed for a specific consensus objective.

In this paper, we exploit a recent *optimistic planning* (OP) algorithm from artificial intelligence to control the agents in discrete time [7]. OP addresses very general optimal control problems, in which a nonlinear system's transitions are evaluated by rewards, and the cumulative reward must be maximized. At each step, OP predicts the system's response to various sequences of actions from the current state, and then chooses an action sequence that appears best. The first action in the sequence is applied, leading to a new state in which the algorithm is applied again, and so on. We express the consensus problem as a set of local optimal control problems for each agent, which are solved with OP. This allows us to deal with general nonlinear agent dynamics, as well as a variety of consensus objectives by simply changing the reward functions: we exemplify full state consensus, partial state consensus (flocking) and leader-based consensus. Coordination between the agent actions is a nontrivial challenge, and a sequential communication procedure [9] is used to address it, in which agents communicate their predicted action sequences in a predefined order.

We illustrate the effectiveness of OP consensus in two problems. The first involces flocking (consensus on velocities) for linear, double integrators under communication range constraints, where OP is compared to the standard flocking algorithm, focusing on the open problem of preserving graph connectivity [11]. In this problem we also study the influence of the parameters of OP consensus. The second problem is the leaderless and leader-based consensus of robotic arms, similar to [12], in which we illustrate how the algorithm deals with nonlinear dynamics. Due to the generality of the approach, theoretical analysis is still open.

The field of consensus is broad, and we direct the interested reader to the surveys [15], [17], [14]. Many approaches are focused on simple or double-integrator agents, whereas OP consensus works for any nonlinear dynamics. Closer to our work is nonlinear flocking and consensus, see for example [18], [21] which handle double integrators with nonlinear acceleration dynamics, [19] which deals with nonholonomic robots, and [12] for Euler-Lagrange dynamics. While these works exploit the characteristics of specific classes of dynamics to derive predefined control laws, OP consensus automatically finds a near-optimal control law, making it adaptable to various dynamics.

It must be noted that OP uses discretized control actions, meaning it can only achieve consensus up to some error given by the discretization accuracy. While this limitation is not fundamental and our approach can be extended to continuous actions, other authors have shown positive theoretical results for consensus with coarsely discretized actions [16].

Since OP is a type of predictive control, our approach relates to distributed model-predictive control [1], and in fact the sequential communication idea is taken from this field [9]. Although nonlinear distributed MPC schemes do exist, in the consensus setting MPC has to our knowledge only been applied to linear systems, e.g. [8], [6].

Next, we introduce OP (Section II), formalize the consensus problem addressed (Section III), and explain the OP consensus approach (Section IV). Sections V and VI show the experiments on double-integrator flocking and robotic arm consensus. Section VII concludes the paper.

## II. Single-agent optimistic planning

Consider an optimal control problem for a deterministic, discrete-time nonlinear system $x_{k+1} = f(x_k, u_k)$ with states $x$ and actions $u$. Each transition is associated with a bounded

The authors are with Université de Lorraine, CRAN, UMR 7039, 2 Avenue de la Forêt de Haye, France and CNRS, CRAN, UMR 7039. {lucian.busoniu, constantin.morarescu} at univ-lorraine.fr

reward $r_{k+1} = \rho(x_k, u_k)$, and the goal is to find a state feedback control policy $h(x)$ that maximizes the infinite-horizon discounted return:

$$V^h(x) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, h(x_k)) \qquad (1)$$

from any state $x$, where $x_0 = x, x_{k+1} = f(x_k, h(x_k))$ for $k \geq 0$. The discount factor $\gamma \in [0, 1)$ ensures the boundedness of the return. The optimal (maximal) value function, denoted $V^* = \max_h V^h$, always exists, is unique, and leads to at least one optimal policy $h^*$. It is also useful to consider an action-dependent optimal value function, the optimal Q-function: $Q^*(x, u) = \rho(x, u) + \gamma V^*(f(x, u))$ [2].[1]

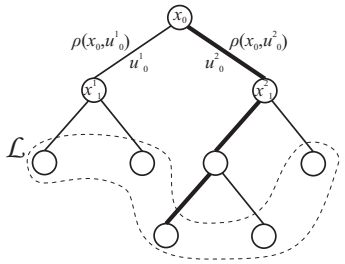



Fig. 1. Illustration of an OP tree $\mathcal{T}$. Nodes are labeled by states, arcs represent transitions and are labeled by the actions taken and the resulting rewards. Subscripts are depths, superscripts index the $M$ possible actions/transitions from a node (here, $M = 2$). The leaves $\mathcal{L}$ are enclosed in a dashed line, while the thick path highlights a possible optimistic sequence.

Optimistic planning (OP) [7] explores a tree representation of the possible action sequences from the current system state, as illustrated in Figure 1. It requires a discrete (or discretized) action space $U = \{u^1, \ldots, u^M\}$, and rewards bounded in $[0, 1]$. In the remainder of this section, we isolate the current time step and by convention relabel the time from $k$ to 0, so that the system state is $x_0$. OP starts with a root node labeled by $x_0$, and iteratively expands $T$ well-chosen nodes. Expanding a node $x$ adds new children nodes containing the next states for all possible discrete actions: $f(x, u^1), \ldots, f(x, u^M)$. Note that a certain state may appear several times in the tree – as many as the number of ways it can be reached from the root. While keeping this in mind, we denote nodes by their label $x$ for simplicity.

Each node $x_d$ at some depth $d$ is reached via a unique path through the tree, associated to a sequence of states $[x_0, \ldots, x_d]$ and a sequence of actions $\boldsymbol{u}(x_d) := [u_0, \ldots, u_{d-1}]$ (note the action sequence is shorter by one element). For a leaf node $x_d \in \mathcal{L}$, the following gives an upper bound on the returns of all infinite action sequences having in common the initial subsequence up to $x_d$:

$$b(x_d) = \sum_{d'=0}^{d-1} \gamma^{d'} \rho(x_{d'}, u_{d'}) + \frac{\gamma^d}{1-\gamma} = \nu(x_d) + \frac{\gamma^d}{1-\gamma}$$

This is because all the rewards at depths larger than $d$ are in $[0, 1]$. For the same reason, $\nu(x_d)$ is a lower bound. The $b$ and $\nu$ values can be efficiently maintained on the tree.

[1]Optimal control problems are often stated so that a cost is minimized, rather than a return being maximized. The two formulations are equivalent.

OP *optimistically* explores the space of action sequences, by always expanding further the most promising sequence: the one with the largest upper bound. This corresponds to expanding the optimistic leaf $x^\dagger = \arg\max_{x \in \mathcal{L}} b(x)$. After the $T$ allowed node expansions are exhausted, a sequence that maximizes the lower bound $\nu(x_d)$ among the leaves $a_d$ is returned, intuitively seen as a safe choice. Typically, the first action in this sequence will be applied to the system. Algorithm 1 summarizes the entire procedure.

**Algorithm 1** Optimistic planning for deterministic systems

1: initialize tree: $\mathcal{T} \leftarrow \{x_0\}$
2: **for** $t = 1, \ldots, T$ **do**
3: find optimistic leaf: $x^\dagger \leftarrow \arg\max_{x \in \mathcal{L}} b(x)$
4: add to $\mathcal{T}$ the nodes $f(x^\dagger, u_j), j = 1, \ldots, M$
5: **end for**
6: **output** return $\boldsymbol{u}(x^*)$, where $x^* = \arg\max_{x \in \mathcal{L}} \nu(x)$

Theoretical analysis shows that OP is a sound algorithm [7]: it returns a near-optimal sequence, while adapting in a certain sense to the complexity of the planning problem at the current state. Intuitively, the algorithm focuses on near-optimal parts of the tree, so when these parts are smaller, it does better. Furthermore, always applying the first action of near-optimal sequences leads to a near-optimal return in closed loop.

## III. Consensus problem

We consider a set of $n$ agents with decoupled nonlinear dynamics $x_{i,k+1} = f_i(x_{i,k}, u_{i,k})$, $i = 1, \ldots, n$. Assuming for simplicity that the dimensionality of $x$, denoted $m$, is the same for every agent (this can easily be relaxed), the goal of achieving consensus on some or all of the state variables can be formalized as:

$$\lim_{k \to \infty} \left| x^c_{i,k} - x^c_{j,k} \right| = 0 \quad \forall i, j = 1, \ldots, n, \forall c \in C$$

where the consensus variables $C \subseteq \{1, \ldots, m\}$. For example, all the variables are synchronized in full-state consensus, while flocking only requires it for the velocities of the agents.

An agent only has a local view: it can receive information only from its neighbors on a (possibly time-varying) interconnection graph $\mathcal{G}_k = (\mathcal{V}, \mathcal{E}_k)$. The set of nodes $\mathcal{V} = \{1, \ldots, n\}$ represents the agents, and the edges $\mathcal{E}_k \subseteq \mathcal{V} \times \mathcal{V}$ are the communication links. Denote by $\mathcal{N}_{i,k} = \{j \mid (i, j) \in \mathcal{E}\}$ the set of neighbors of node $i$ at step $k$. How the graph varies depends on the problem at hand, and an example will be provided in Section V.

A few other graph theory concepts will be useful. A path through a generic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a sequence of nodes $i_1, \ldots, i_L$ so that $(i_l, i_{l+1}) \in \mathcal{E}, 1 \leq l < L$. The graph is connected if there is a path between any pair of nodes $i, j$. The graph's adjacency matrix is $A \in \{0, 1\}^{n \times n}$, with $A_{i,j} = 1$ iff $(i, j) \in \mathcal{E}$; and the degrees vector is: $\Delta \in \{0, \ldots, n\}^n$, $\Delta_i = \sum_j A_{i,j} = |\mathcal{N}_i|$. The weights can generally take any positive value but we only need unitary weights. The graph is undirected if $A$ is symmetric, and directed otherwise (if

some of the connections are one-way). Graph connectivity is crucial for consensus, and all algorithms require it in one way or another [15], [17], [14].

## IV. Optimistic planning for consensus

Next, we present the OP-based approach to the consensus problem in Section III. At every time step $k$, a local optimal control problem is defined for each agent $i$, using information locally available to it. The goal in this problem is to align the consensus states with those of the neighbors $\mathcal{N}_{i,k}$, and if the connection graph is varying, to also maintain connectivity with them. OP is used to near-optimally solve this control problem, and the first action of the sequence returned is applied by each agent. Then the system evolves, and the procedure is applied at the next step, for the new states and possibly changed graph.

To construct its optimal control problem, each agent needs to know the predicted behavior of its neighbors. Here, agents will exchange the near-optimal action sequences returned by OP. Because the agents must act at the same time, how they exchange predictions is nontrivial. The simplest solution would be to use the neighbors' predictions at the previous step. However, since the neighbors revise their solutions in the meantime, their actions may change, which leads to a *coordination* problem. Coordination is a difficult challenge in multi-agent systems, see e.g. [3], and is typically solved in model-predictive control by explicit, iterative negotiation over successive local solutions [13]. However, in consensus it is unlikely that the agents can afford to repeatedly communicate and reoptimize their solutions at every step.

Therefore, we adopt a sequential communication procedure in which agents optimize once per step, similar to the procedure for distributed MPC in [9]. Each agent needs to know its index $i$ as well as the indices of its neighbors (one way to ensure this is an initial, centralized assignment of indices to the agents). Agent $i$ waits until the neighbors $j$ with $j < i$ have found their action sequences. These agents communicate their sequences to $i$. For $j > i$, agent $i$ heuristically assumes they will follow their *previous* sequences. Agent $i$ exploits all these sequences to predict the likely evolution of its neighbors, optimizing its own behavior while coordinating with this evolution. It then sends its own, newly computed sequence to neighbors $j > i$.

To formalize the algorithm, recall first that the planner of some agent $i$ returns at step $k$ an action sequence $\boldsymbol{u}_i^k = [u_{i,k}^k, u_{i,k+1}^k, ..., u_{i,k+d-1}^k]$ (see Algorithm 1, remembering that time $k$ is relabeled there to 0). The superscript $k$ is needed to differentiate between sequences found at different time steps, since they may have different actions at corresponding positions (e.g., $u_{i,k+1}^k$ may be different from $u_{i,k+1}^{k+1}$). Consider now a specific agent $i$. At every step $k$, it receives the states $x_{j,k}$ of its neighbors $j \in \mathcal{N}_{i,k}$. For neighbors $j \in \mathcal{N}_{i,k},\ j < i$, agent $i$ directly receives their sequence at $k$ and uses this as an estimation of their future behavior: $\widehat{\boldsymbol{u}}_j^{i,k} = \boldsymbol{u}_j^k$. For $j \in \mathcal{N}_{i,k},\ j > i$, it uses their *previously* chosen sequences $\widehat{\boldsymbol{u}}_j^{i,k} = [u_{j,k}^{k-1}, ..., u_{j,k-1+d-1}^{k-1}]$, where the first action $u_{j,k-1}^{k-1}$ has been discarded because it has already been applied and is no longer informative. Note we use the superscript $i,k$ to highlight variables specific to the control problem constructed by agent $i$ at time $k$.

Using this information, agent $i$ applies OP to an optimal control problem with dynamics $f_i$ and the reward function:

$$\rho_d^{k,i}(x_{i,d}, u_{i,d}) = \frac{1}{|\mathcal{N}_{i,k}|} \sum_{j \in \mathcal{N}_{i,k}} \Big[ -\alpha (x_{j,d} - x_{i,d})^\top W (x_{j,d} - x_{i,d}) - \beta \cdot \begin{cases} 1 & \text{if connection } (i,j) \text{ lost at } d \\ 0 & \text{otherwise} \end{cases} \Big] \tag{2}$$

The first term deals with alignment of the consensus states and the second with maintaining connectivity, with $\alpha$ and $\beta$ weighing the relative importance of these terms. Typically, $\beta$ will be larger than the maximum range of the alignment term, so that connectivity is given priority. Matrix $W \in \mathbb{R}^{m \times m}$ is diagonal and satisfies $W_{c,c} = 0$ if $c \notin C$. Its nonzero terms are used to weigh the relative importance of achieving consensus on the different states. When the graph is fixed, the second term is always 0 so the values of $\alpha, \beta$ become irrelevant. Note that state differences should be saturated at a sufficiently large value to keep the reward bounded, and for the sake of OP the total reward should then be scaled and translated in the interval $[0, 1]$. With modifications to the reward function, additional objectives could be encoded, such as formation maintenance or collision avoidance.

The time index $d$ is used to reemphasize the fact that time is equivalent to depth in the planning tree, and that these rewards are "virtual" in the planning problem solved at time $k$. The neighbor trajectories are simulated using their communicated action sequences, $x_{j,d+1} = f_j(x_{j,d}, \widehat{u}_{j,k+d}^{i,k})$. Depth $d$ may exceed the length of the available action predictions; when that happens those neighbors' actions are set to 0. In the implementation, the agents could also exchange the predicted state sequences instead of the actions, which at some extra communication cost avoids resimulating the neighbor's transitions up to the prediction length.

The reward function $\rho_d^{k,i}$ is *time-varying* (hence the subscript $d$), due to the dependence on the neighbors' trajectories. The problem solved by agent $i$ is therefore more general than the standard time-invariant problem introduced in Section II. Nevertheless, OP does not rely on time invariance and so can deal with this generalization without changes to its theoretical guarantees.

Algorithm 2 summarizes the resulting consensus protocol for generic agent $i$. The main advantage of this algorithm is the generality of the agent dynamics and consensus objectives it can address. This generality comes at the cost of communicating predictions, introducing a dependence of the performance on the action discretization, and a relatively computationally involved algorithm. Our empirical study below will illustrate that a meaningful tradeoff between discretization resolution and performance can be achieved, and that limiting the length of the communicated sequences to a small value does not reduce performance. As to computational cost, the time complexity of each individual OP

**Algorithm 2** OP consensus at agent $i$

set initial action prediction $\boldsymbol{u}_i^{-1}$ to an empty sequence
**for** $k = 0, 1, 2, \dots$ **do**
  exchange states at $k$ with all $j \in \mathcal{N}_{i,k}$
  send $\boldsymbol{u}_i^{k-1}$ to $j < i$, receive $\widehat{\boldsymbol{u}}_j^{i,k}$ from $j > i$
  wait until new sequences $\widehat{\boldsymbol{u}}_j^{i,k}$ received from all $j < i$
  run OP with reward (2), obtaining $\boldsymbol{u}_i^k$
  send $\boldsymbol{u}_i^k$ to $j > i$
  execute first action $u_{i,k}^k$ and remove it from $\boldsymbol{u}_i^k$
**end for**

application is between $O(T \log T)$ and $O(T^2)$ depending on the planning problem complexity, see [5]. The overall complexity for all agents, if they run OP in parallel as soon as the necessary neighbor predictions become available, is larger by a factor equal to the length of the longest path from any $i$ to any $j > i$. Depending on the current graph this length may be significantly smaller than $n$.

## V. Results for double-integrator flocking

In a first set of experiments, we will apply the framework developed above to flocking. In this problem, the state of each agent $i$ consists of a position $p_i \in \mathbb{R}^{m/2}$ and a velocity $v_i \in \mathbb{R}^{m/2}$, $x_i = [p_i^\top, v_i^\top]^\top$. Connectivity is time-varying, and two agents $i, j$ are directly connected if they are within a communication range $P$, leading to $\mathcal{E}_k = \{(i,j) \mid i \neq j, \|p_{i,k} - p_{j,k}\| \leq P\}$. The goal is consensus on the velocities, achieved by setting $C = \{m/2+1, \dots, m\}$.

So far the agents can have any dynamics. The most commonly used dynamics are double integrators [14], which in discrete time are, using Euler discretization:

$$p_{i,k+1} = p_{i,k} + T_\mathrm{s} v_{i,k}, \quad v_{i,k+1} = v_{i,k} + T_\mathrm{s} u_{ik}$$

with sampling time $T_\mathrm{s}$ (note however that results exist for nonlinear agents, e.g. [19], [18], [21]). A classical flocking algorithm is:

$$u_{i,k} = \alpha_k \sum_{j \in \mathcal{N}_{i,k}} (v_{j,k} - v_{i,k}) \tag{3}$$

where the gain $\alpha_k < 1/\max_i \Delta_{i,k}$. With this algorithm, *if the graph remains connected* (in a certain weak sense, over time), flocking to the average of the initial agent velocities, denoted $v^*$, will be achieved [15].

An important unsolved problem, even for this simple case, is ensuring connectivity. Most often, connectivity is simply assumed. One interesting exception which does provide connectivity guarantees is [11]. These results are for continuous-time double-integrators: $\dot{p}_i(t) = v_i(t)$, $\dot{v}_i(t) = u_i(t)$ controlled with the continuous-time version of (3): $u_i(t) = \sum_{j \in \mathcal{N}_i(t)} (v_j(t) - v_i(t))$. Then, for the graph $\mathcal{G}_0$ corresponding to the initial positions of the agents, a quantity called robustness is defined, denoted $P^\mathrm{r}$ and intuitively understood as the maximal amount by which all inter-agent distances can grow while the graph still remains connected. A so-called minimal robust subgraph $\mathcal{G}^\mathrm{r}$ is also defined, obtained from $\mathcal{G}_0$ by actually increasing the distances by $P^\mathrm{r}$; this graph contains the essential edges that should never be broken. The result states that, if the initial velocity disagreement vector $\delta_0 = [v_{1,0}^\top - v^{*\top}, \dots, v_{n,0}^\top - v^{*\top}]^\top$ satisfies $\|\delta_0\| \leq \lambda_2^\mathrm{r} P^\mathrm{r}/\sqrt{2}$, connectivity is never broken and flocking is achieved. Here $\lambda_2^\mathrm{r}$ is the algebraic connectivity of $\mathcal{G}^\mathrm{r}$, see e.g. [15]. Ref. [11] empirically illustrated that, while conservative, the result is meaningful: the algorithm still achieves flocking if the initial disagreement is somewhat larger than the critical value, but if it is too large connectivity is broken and flocking fails. Since algorithm (3) is the discretized variant of the one in [11], we expect similar conditions to hold (possibly stricter because discretization reduces control freedom).

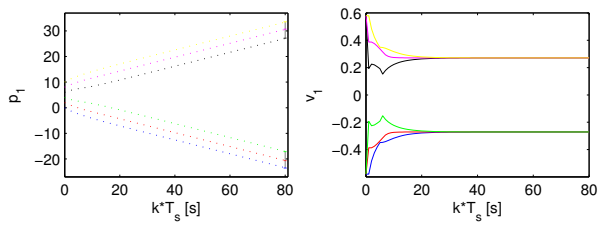
There is no reason for these limitations beyond the specific algorithm used: if their control constraints allow the agents to reach trajectories that maintain connectivity, then flocking is still possible. So our first goal in this section is to empirically verify if, by using OP consensus, we achieve flocking when the robustness condition is sufficiently violated to cause the failure of the classical algorithm. Secondly, we perform a thorough numerical study of the influence of the OP parameters on flocking performance.

### A. Flocking results

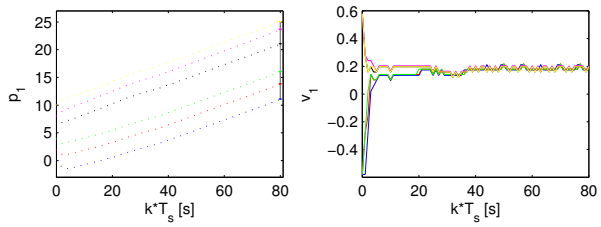
Single- and two-dimensional double integrators are considered. In the reward function, all the weights of the consensus states are set to 1, and $\alpha = \beta = 0.5$ (the velocity disagreement term is normalized into the range $[0, 1]$, so the penalty on disconnection is always larger than that on disagreement).

In the 1D case, six agents are initialized on an equidistant grid with a spacing of 2. Their communication range is 5, so the initial graph has some redundant connections. The sampling time is $T_\mathrm{s} = 1\,\mathrm{s}$. The initial velocities are initialized at $v_0$ for the "top" three agents, and $-v_0$ for the "bottom" three. This is very similar to the example considered in [11]. From the robustness condition $\|\delta_0\| \leq \lambda_2^\mathrm{r} P^\mathrm{r}/\sqrt{2}$, a critical initial velocity $v^\mathrm{r} \approx 0.23$ is found. When $v_0 = v^\mathrm{r}$, the algorithm (3) works (we do not show a graph since the result is trivial). However, when $v_0 = 2.5 v^\mathrm{r}$, the classical algorithm already fails, as shown in Figure 2(a).[2] The gain in (3) is $\alpha = 1/n$, which is a safe lower bound on $1/\max_i \Delta_{i,k}$, which cannot be used because it depends on the maximal degree, a global quantity unknown to the agents. Turning to the OP method, it easily achieves flocking, up to errors due mainly to the discretized actions, see Figure 2(b). The discretized action set contains in this case the 7 values $\{-0.3, -0.122, -0.039, 0, 0.039, 0.122, 0.3\}$, and the planning budget is $T = 200$ node expansions. The discount factor $\gamma$ is set to 0.98 for all the experiments of this paper, which leads to considering long-term rewards with a significant weight.

[2] The initial velocity disagreement is thus $2.5\sqrt{6} \approx 6$ times larger than the critical value. This roughly corresponds to the continuous-time results of [11] who found that when the disagreement is 5 times the critical value, the agents fail to flock.




(a) Classical flocking.




(b) OP flocking.

Fig. 2. Results in the 1D case, when $v_0 = 2.5v^{\mathrm{r}}$. The left graph shows the velocities of the agents over time, and the right graph their trajectories. The final configuration of the agents is also shown, with the agents represented as dots and the communication graph with gray lines.

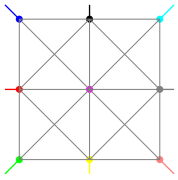
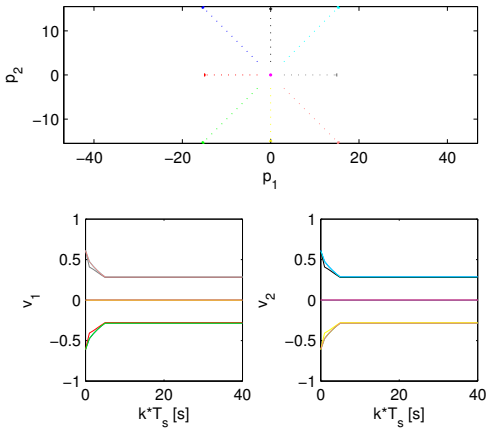
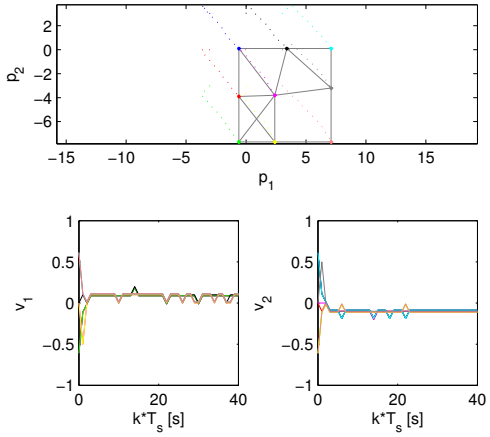


Fig. 3. 2D case, initial configuration: agents are colored dots and their initial velocity is shown in thick colored lines. The graph is shown in gray.




(a) Classical flocking.




(b) OP flocking.

Fig. 4. Results in the 2D case, when $v_0 = 1.5v^{\mathrm{r}}$. The bottom two graphs show the two velocity components over time. The top graph shows the 2D evolution of the agent positions in dotted lines, with the final configuration of the agents included.

In the 2D case, 9 agents are arranged on an equidistant grid. The range is 5 and the grid step is 3. Each agent initially moves away from the grid center, see Figure 3, and each initial velocity component is set to $v_0$. Figure 4 shows the performance of classical flocking ($\alpha = 1/n$) and OP flocking when $v_0 = 1.5v^{\mathrm{r}}$, with $v^{\mathrm{r}}$ the critical value found from the robustness condition.[3] As in the 1D case, classical flocking fails while OP succeeds up to discretization errors. Here the action discretization was $\{-0.5, -0.10, 0.1, 0.5\} \times \{-0.5, -0.10, 0.1, 0.5\}$ (the same values for both axes), the budget was $T = 200$.

We conclude that, by acting less conservatively than the classical algorithm, OP consensus allows the agents to maintain connectivity and thus achieve flocking from a wider range of initial conditions. If sufficiently large discrete actions are available, OP should be able to handle correspondingly very large initial disagreements.

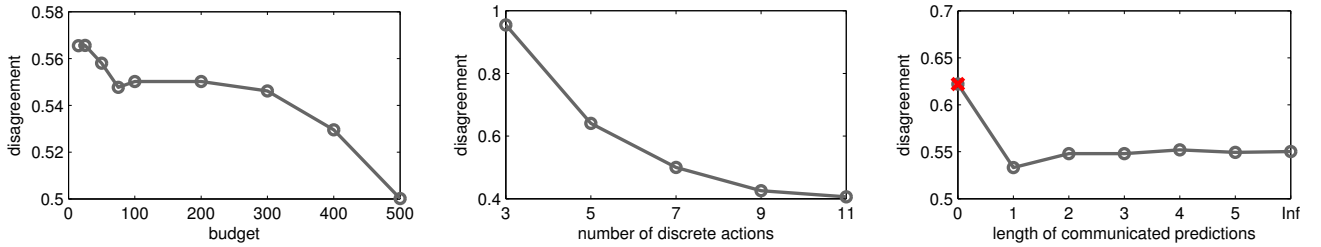
### B. Effects of OP tuning parameters

For the remainder of the experiments, the 1D case with $v_0 = 2.5v^{\mathrm{r}}$ is considered. To characterize performance in each experiment with a single number, an inter-agent velocity disagreement is computed at every step: $\tilde{\delta}_k = \sum_{i<j} \|v_{i,k} - v_{j,k}\|$, and the average of $\tilde{\delta}_k$ across all steps in the trajectory is reported.

*a) Budget:* First the algorithm is run with different expansion budgets, $T = 15, 25, 50, 75, 100, 200, \ldots, 500$. As shown in Figure 5(a) and as expected from the theoretical guarantees of OP, performance largely increases with $T$, although monotonicity is not guaranteed, as exemplified on the graph around $T = 100$.

*b) Action discretization:* To investigate the influence of the action discretization, we run OP flocking with several discretizations, having respectively $3, 5, 7, 9$, and $11$ elements; the discrete values are symmetrical and logarithmically spaced around 0, in the range $[-0.3, 0.3]$ (see Section V-A for the 7-value set as an example). The budget is set to a large value $T = 500$ to allow the trees to be well developed even for the larger discretizations. Figure 5(b) presents the results. Performance improves as the discretization becomes finer, and the influence on the performance is much larger than that of the budget above, although the gain tapers off for the finest discretizations. This suggests the strategy of setting some large enough value for the budget, and then selecting

[3]It is interesting to note that classical flocking fails in this problem even when $v_0 = v^{\mathrm{r}}$; this is due to the discretization in time, and more specifically to the conservativeness of the $\alpha$ value.




(a) Budget. (b) Action discretization. (c) Maximal prediction length ("Inf" means it is not limited).

Fig. 5. Influence of OP consensus parameters. An X marks lost connectivity in that experiment. Note the changing vertical scale.

the largest discretization that computational resources allow.

*c) Length of action predictions:* Since the communication overhead is important, we are interested how OP consensus works when the agents only exchange limited-length action plans. Thus for $T = 200$ and the 7-action grid we repeat the experiment while increasing the length of communicated predictions from 0, i.e. no communication, gradually to 5, and then letting them be unlimited (recall that once the predicted sequences are exhausted, the agents just simulate their neighbors assuming they will apply action 0). Figure 5(c) indicates that communicating plans is indeed necessary; without them, connectivity is lost. Nevertheless, plans of length 1 are already sufficient for good performance. The lack of improvement (and even slight decrease) in performance for larger lengths is more surprising; presumably due to the receding-horizon nature of the algorithm, making errors in the predictions over the longer term is not too important.

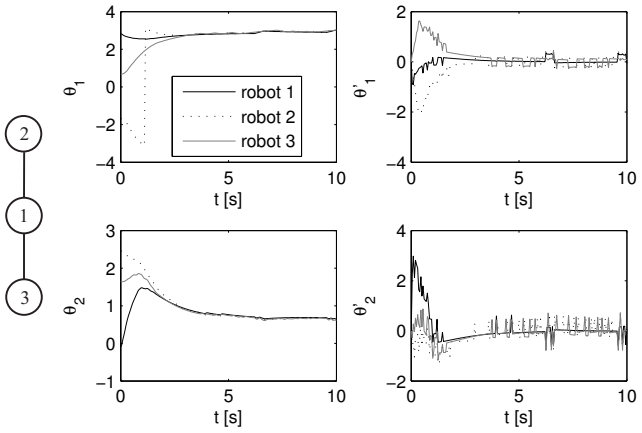



Fig. 6. Leaderless consensus of multiple robotic arms. Left: communication graph. Right: controlled trajectories.

## VI. Results for nonlinear robotic arms

To investigate the performance of the algorithm for nonlinear agent dynamics, we apply it to the consensus of two-link robotic arms operating in a horizontal plane. The state variables for each agent are the angles and angular velocities of the two links, $x_i = [\theta_{i,1}, \dot{\theta}_{i,1}, \theta_{i,2}, \dot{\theta}_{i,2}]$, and the actions are the torques of the motors actuating the two links $u_i = [\tau_{i,1}, \tau_{i,2}]$. The model is standard so we omit the details and just note that the sampling time is $T_\mathrm{s} = 0.05\,\mathrm{s}$; the other parameters can be found in [4]. Applications of this type of consensus problem include decentralized manipulation and teleoperation.

First, we consider regular consensus starting from random initial angular positions with zero initial velocities, see Figure 6. Three robots are connected on the fixed communication graph shown on the left of the figure. The goal is to achieve consensus on all states. The weights are set to $W = \mathrm{diag}[1, 0.1, 1, 0.1]$, so the angles are given higher priority. The discretized action set is $\{-1.5, 0, 1.5\} \times \{-1, 0, 1\}$, and the budget of each agent is $T = 400$. Consensus is easily achieved by the algorithm.

The second task is consensus with a leader. In this case, leader agent 1 is solving its own, single-agent control problem, and the other agents $2, \ldots, n$ must synchronize with it, so the graph is directed without any arcs pointing to the leader. Algorithm 2 is applied only at agents $i > 1$, whereas the leader can use any control algorithm with the restriction that it should provide predictions of the leader's behavior to the other agents. A natural choice is single-agent OP (Algorithm 1).

Here, the control problem solved by the leader is tracking a circle with the end-effector. The circle has center $(0.35, 0.35)$, radius $0.2$ and must be tracked at angular velocity $2\pi/3\,\mathrm{rad/s}$. The leader's rewards penalize the deviations of the end-effector from the reference trajectory in cartesian coordinates:

$$r_{1,k+1} = -(X_k - X_k^{\mathrm{ref}})^2 - (Y_k - Y_k^{\mathrm{ref}})^2$$

where $X$ and $Y$ are the coordinates and superscript ref denotes the reference trajectory. The planning budget and action discretizations are for all agents the same as in the simple consensus problem, and $W$ also stays unchanged. The communication graph is shown in Figure 7, left, with the resulting trajectories on the right. Agents 2 and 3 reach consensus with the leader 1 after about $2.5\,\mathrm{s}$ and thereafter track the leader's trajectory (which in turn tracks the reference circle, although this is not as easily visible in the figure).

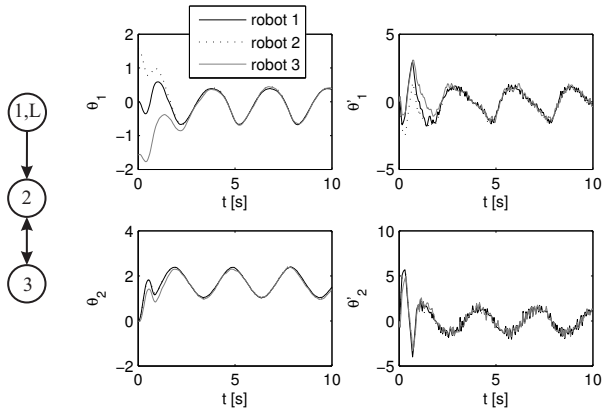



Fig. 7. Leader-based consensus of multiple robotic arms. Agent 1 is the leader.

## VII. Conclusions

We have introduced a novel consensus approach based on optimistic planning, a nonlinear predictive control algorithm from the field of artificial intelligence. This *optimistic planning consensus* has shown good results for linear, as well as nonlinear agent dynamics, and several types of consensus objectives. Among the parameters of the algorithm, the discretization of the action space has the greatest influence, and future effort should be directed to eliminate this discretization step. A continuous-action version of single-agent optimistic planning is under development, and some continuous-action planning algorithms already exist [10].

## References


[1] A. Bemporad and D. Barcelli, "Decentralized model predictive control," in *Networked Control Systems*, ser. Lecture Notes in Control and Information Sciences, A. Bemporad, M. Heemels, and M. Johansson, Eds. Springer, 2010, vol. 406, pp. 149–178.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, 2007, vol. 2.

[3] L. Buşoniu, R. Babuška, and B. De Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics. Part C: Applications and Reviews*, vol. 38, no. 2, pp. 156–172, 2008.

[4] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Approximate dynamic programming with a fuzzy parameterization," *Automatica*, vol. 46, no. 5, pp. 804–814, 2010.

[5] L. Buşoniu and R. Munos, "Optimistic planning for Markov decision processes," in *Proceedings 15th International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, ser. JMLR Workshop and Conference Proceedings, vol. 22, La Palma, Canary Islands, Spain, 21–23 April 2012, pp. 182–189.

[6] G. Ferrari-Trecate, L. Galbusera, M. Marciandi, and R. Scattolini, "Model predictive control schemes for consensus in multi-agent systems with single- and double-integrator dynamics," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2560–2572, 2009.

[7] J.-F. Hren and R. Munos, "Optimistic planning of deterministic systems," in *Proceedings 8th European Workshop on Reinforcement Learning (EWRL-08)*, Villeneuve d'Ascq, France, 30 June – 3 July 2008, pp. 151–164.

[8] T. Keviczky and K. Johansson, "A study on distributed model predictive consensus," in *Proceedings 17th IFAC World Congress (IFAC-08)*, Seoul, Korea, 6–11 July 2008, pp. 1516–1521.

[9] J. Liu, X. Chen, D. M. de la Peña, and P. D. Christofides, "Sequential and iterative architectures for distributed model predictive control of nonlinear process systems," *American Institute of Chemical Engineers (AIChE) Journal*, vol. 56, no. 8, pp. 2137–2149, 2010.

[10] C. Mansley, A. Weinstein, and M. L. Littman, "Sample-based planning for continuous action Markov decision processes," in *Proceedings 21st International Conference on Automated Planning and Scheduling*, Freiburg, Germany, 11–16 June 2011, pp. 335–338.

[11] S. Martin and A. Girard, "Sufficient conditions for flocking via graph robustness analysis," in *Proceedings 49th IEEE Conference on Decision and Control (CDC-10)*, Atlanta, US, 15–17 December 2010, pp. 6293–6298.

[12] J. Mei, W. Ren, and G. Ma, "Distributed coordinated tracking with a dynamic leader for multiple euler-lagrange systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1415–1421, 2011.

[13] R. R. Negenborn, B. De Schutter, and H. Hellendoorn, "Multi-agent model predictive control for transportation networks: Serial versus parallel schemes," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 3, pp. 353–366, 2008.

[14] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[15] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[16] C. D. Persis and P. Frasca, "Robust self-triggered coordination with ternary controllers," 2012, arXiv preprint 1205.6917.

[17] W. Ren and R. W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control: Theory and Applications*, ser. Communications and Control Engineering. Springer, 2008.

[18] H. Su, G. Chen, X. Wang, and Z. Lin, "Adaptive second-order consensus of networked mobile agents with nonlinear dynamics," *Automatica*, vol. 47, no. 2, pp. 368–375, 2011.

[19] H. Tanner, A. Jadbabaie, and G. Pappas, "Flocking in teams of nonholonomic agents," in *Cooperative Control*, ser. Lecture Notes in Control and Information Sciences, V. Kumar, N. Leonard, and A. Morse, Eds. Springer, 2005, vol. 309, pp. 458–460.

[20] N. Vlassis, *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*, ser. Synthesis Lectures in Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2007.

[21] J. Zhou, X. Wu, W. Yu, M. Small, and J. Lu, "Flocking of multi-agent dynamical systems based on pseudo-leader mechanism," *Systems & Control Letters*, vol. 61, no. 1, pp. 195–202, 2012.