Decentralized Estimation of the Minimum Strongly Connected Subdigraph for Robotic Networks with Limited Field of View

Cosimo Federico Ardito, Donato Di Paola, and Andrea Gasparri

Abstract—In this work we focus on the topology control problem for robotic networks. In particular, we assume agents to be equipped with limited field of view sensors. As a consequence, directed graphs are required to model the robot-to-robot interaction. This significantly limits the applicability of algorithms developed for undirected graphs. In that view, we propose an auction-based solution for the decentralized estimation of an approximated minimum (in terms of number of links and in terms of a global cost function) strongly connected directed graph. This represents the first step towards the development of a connectivity maintenance framework for directed graphs. A theoretical analysis along with numerical simulations are provided to show the effectiveness of the proposed approach.

I. INTRODUCTION

In recent years, robotic networks have gained the attention of several researchers in the robotics community. The interest in this area is mostly motivated by the wide range of possible applications, ranging from area coverage [1] to sweeping operations [2]. In order to achieve these collaborative tasks, robots are required to cooperate, thus the maintenance of the network connectivity becomes a relevant problem.

A great amount of work can be found in the literature concerning the connectivity maintenance of the network topology of multi-agent systems [3], [4], [5], [6], [7]. A very common approach is based on the idea of maximizing the algebraic connectivity. To this end, some authors have proposed decentralized control laws based on the explicit estimation of the algebraic connectivity and/or its related eigenvector [8], [9], [10]. Other authors have designed control laws based on artificial potential fields. In [11], the authors propose a distributed framework in which the connectivity of a given initially connected network topology is preserved using an hybrid approach. This framework controls the deletion of links through a market-based protocol and maintains the existing links utilizing a motion control law based on nearest-neighbor potential fields. Of particular interest for applications are those works in which the connectivity maintenance is jointly performed with a coordination task, such as in [12], in which both swarming and connectedness are guaranteed by means of suitable artificial potential functions, or in [13], in which both flocking and connectivity of the agents is guaranteed at the same time.

A common assumption of those works is that the graph describing the interaction among the robotic units is undirected. This implies agents to have uniform and isotropic sensing capabilities. Therefore, these approaches cannot be applied for all those scenarios where agents are assumed to be equipped with limited Field Of View (FOV) sensors.

In this paper we extended the results presented in [14], where the problem of decentralized connectivity control of a network of robotic agents equipped with heterogeneous and limited field of view sensors is addressed. This approach, inspired by [15], [16], points to find a suitable structure, which is a critical subgraph on which links cannot be removed any more without losing the connectivity property. This approach can be viewed as a first step towards the design of a decentralized control law which allows a robot to move so as to never break links part of the critical structure. The improvement made in this work, compared to the original framework [14] is twofold: on the one hand an auction-based approach ensures consistency of information across the network guaranteeing a good performance of the algorithm in terms of computational complexity, on the other hand in the directed interaction graph each link is associated to a weight. This takes into account a certain cost function, such as the geometric distances among agents, which is useful to minimize during the operations of the network.

II. PRELIMINARIES

Consider a heterogenous robotic network composed of $n$ agents with limited FOV where heterogeneity is meant in terms of sensing capability. The following assumptions are made on the robotic network.

Assumption 1: For each robotic agent $i$:

i) There is a sensory system represented by a circle sector with angle $\alpha_i$ and radius $\rho_i$ attached to it.

ii) There is a wireless bi-directional communication system with a limited radius $\delta$, equal for all agents.

iii) The communication radius is larger than the sensing radius for all robots, that is: $\delta \geq \rho_i \quad \forall i \in \{1, \ldots, n\}$.

iv) There is a common heading (provided for instance through a compass) by which the global orientation $\theta_i(t) \in [-\pi, \pi]$ is defined.

v) There is no awareness of its global position $p_i(t) \in \mathbb{R}^2$ nor of any other agent at time $t$.

vi) There is a unique ID associated to it by which all agents can be distinguished for sensing and communication.
purposes.

Let us now introduce the concept of communication and sensing graphs for the interaction modeling of a robotic network with limited FOV.

Definition 1 (Communication Graph): Let the communication graph be the undirected graph $\mathcal{G} = \{ V, \mathcal{E} \}$, with $V$ the set of nodes describing the robotic agents and $\mathcal{E} \subseteq V \times V$ the set of pairs of nodes, called links, where a link $(i, j) \in \mathcal{E}$ exists if the robot $i$ is able to communicate with robot $j$.

Definition 2 (Sensing Graph): Let the sensing graph be the directed graph $D = \{ V, \mathcal{E}, w \}$, where $V$ is the set of nodes, $\mathcal{E} \subseteq \hat{V} \times \hat{V}$ is a set of links, and $w : \mathcal{E} \rightarrow \mathbb{R}^+$ is the weight function which associates a real positive number $w_{ij}$ to each link $(i, j)$. A link $(i, j) \in \mathcal{E}$ exists if robot $i$ is able to sense robot $j$. Self-links $(i, i)$ are not allowed.

Two remarks are now in order:

a) From the bi-directionality assumption on the communication system it follows that the existence of a link $(i, j)$ implies the existence of the link $(j, i)$.

b) From the assumption $\delta \geq \rho_i$, $\forall i \in [1, \ldots, n]$ it follows that the existence of a link $(i, j) \in \mathcal{E}$ in the sensing graph implies the existence of the link $(i, j) \in \mathcal{E}$ in the communication graph. Thus, given a sensing graph the related communication graph can be always established.

Let us now introduce the concepts of strongly connectivity and critical spanning subgraph.

Definition 3 (Strong Connectivity): A given digraph $D = \{ V, \mathcal{E} \}$ is strongly connected if, for any pair of node $(i, j) \in V$ there exists two direct paths $^1 p_{ij}$ and $p_{ji}$, where $p_{ij}$ is the path from node $i$ to node $j$, and $p_{ji}$ from node $j$ to node $i$.

Definition 4 (Critical Spanning Subgraph): For a given digraph $D = \{ V, \mathcal{E} \}$, a critical spanning subgraph (CSS) is a digraph $D^* = \{ V, \mathcal{E}^*, w \}$, where $\mathcal{E}^* \subseteq \mathcal{E}$, such that it is strongly connected and has the minimum number of links $L = |\mathcal{E}^*|$ and the minimum total weight $W = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}$.

Since the implicit goal of any CSS search algorithm is to minimize the number of outgoing links for each node, it is necessary that each agent always chooses a multi-hop path, if it exists, instead of the direct link to each one of its neighbors. For this, we consider the following assumption.

Assumption 2: The weight function $w : \mathcal{E} \rightarrow \mathbb{R}^+$ satisfies the triangle inequality, that is given tree agents $i, j$, and $k \in V$ and the direct links $(i, j), (i, k), (k, j) \in \mathcal{E}$ with the associated weights $w_{ij} \leq w_{ik} + w_{kj}$. This implies that $w_{ij} = w_{ji}$.

Figure 1(a) shows a network of six agents with limited FOV, while Figure 1(b) shows the digraph $D$ deriving from sensing interactions among robots and Figure 1(c) show a possible Critical Spanning Subgraph.

Note that for given a directed graph the CSS might not be unique as there might be a set of optimal solutions $\mathcal{P}$ which are not strictly dominated by any other solution in $\mathcal{P}$, that is their value cannot be improved in one dimension (total weight / number of links) without being worsened in the other (number of links / total weight). This set $\mathcal{P}$ is called Pareto front. Figure 2 shows an example of Pareto front for a connected digraph $D$ with $n = 4$ nodes. In particular, the CSS in Figure 2(b) is the one with the minimum total weight $W = 8$ and $L = 5$ links, while the CSS in Figure 2(c) has $L = 4$ links and the minimum total weight $W = 16$. None of the two solutions is strictly dominated by the other.

$^1$A path $p_{ij}$ between two nodes $i, j \in V$ is a sequence of links, in a directed graph, of the form $p_{ij} = \{ (i, i_1), (i_1, i_2), \ldots, (i_{h-1}, i_h), (i_h, j) \}$.
III. Decentralized Estimation of the Critical Subgraph

In this section, we first introduce the key idea of proposed approach to estimate the Critical Spanning Subgraph (CSS), and then we give a detailed description of the proposed Centralized Estimation of Critical Subgraph (DECS) algorithm.

The objective of the DECS algorithm is to compute in a decentralized fashion a spanning subgraph of a given digraph so that the (strong) connectivity is preserved. Formally, given the strongly connected weighted digraph $D = \{V, E, w\}$, DECS finds an approximate Critical Spanning Subgraph (CSS), denoted as $D'$, of the original digraph $D$. Note that, no global knowledge of the digraph is available to the agents, thus the development of a decentralized algorithm is required.

To the best of the authors knowledge, there is no algorithm in the literature for the (decentralized) detection of a CSS able to minimize both the number of links and the total weight. However, there exist results which show the CSS problem detection to be NP-hard in the case only the number of links minimization is considered [17], [18]. Indeed, all these works cope with the detection of an approximated (suboptimal) CSS.

A. The main idea

The key idea of the algorithm is to compute a set of rooted directed spanning tree each one rooted at one node in a decentralized way. In particular, each agent attempts to compute the following spanning tree:

Definition 5 (Direct Minimum Spanning Tree): Consider a weighted digraph $D$ with a weight $w_{ij}$ associated to each link $(i,j) \in E$. We define, for the node $i \in V$, the rooted Directed Minimum Spanning Tree (DMST) $T_i = \{V, E_i^*, \}$, where $E_i^* \subseteq E$, the rooted tree which minimizes the sum of $w_{ij}$ for all $(i,j) \in E_i^*$.

The DMST can be defined as a graph which connects, without any cycle, all nodes with $n-1$ links. Several algorithms can be found in the literature to compute the DMST from a given digraph. Efficient centralized algorithms have been proposed by Chu and Liu [19], Edmonds [20], and Bock [21]. A distributed algorithm has been proposed by Humblet [22]. In our work, we resort to this distributed algorithm for the computation of the of outgoing links associated to the DMST of a given agent in the Outgoing Link Detection (OLD) procedure.

B. The DECS algorithm

The Decentralized Estimation of Critical Subgraph (DECS) is an iterative algorithm composed of two phases, that is “link selection process” and “consensus for deletion”. In the first phase, each agent computes the set of its outgoing links belonging to the DMST and possibly selects one of them as a candidate for the deletion, while in the second phase, a single redundant link is selected among all candidates for deletion. The algorithm stops if deletions are no longer allowed.

Algorithm 1 - DECS - Phase 1 for agent $i$ at iteration $k$

1: $y_i(k) = 0$
2: $h_i(k) = 0$
3: $e_i(k) = 0$
4: $b_i(k) = [0, i]^T$
5: $[w_i(k), y_i(k)] = \text{OLD}(w_i(k-1))$
6: $c_i(k) = \sum_j I(w_{ij}(k) > 0)$
7: $h_{ij}(k) = I(y_{ij}(k) < c_i(k)), \forall j \in V$
8: if $h_i \neq 0$ then
9: $e_i(k) = \text{argmax}_j h_{ij}(k) \cdot w_{ij}(k)$
10: $b_i(k) = w_i.e_i$
11: end if

For the execution of the DECS algorithm each agent is required to store, at each iteration $k$, two vectors of length $n$ and one vector of length 2:

- $w_i(k) \in \mathbb{R}^n$, $w_i(k) = [w_{i1}(k), \ldots, w_{in}(k)]^T$;
- $y_i(k) \in \mathbb{N}^n$, $y_i(k) = [y_{i1}(k), \ldots, y_{in}(k)]^T$;
- $b_i(k) = [w_i.e_i, i]^T$.

The weight vector $w_i(k)$ describes the weights of the outgoing links of agent $i$ at iteration $k$. The vector $w_i(k)$ is initialized with the weights of outgoing links for each agent $i$, as in the original the graph $D$. The link counter vector $y_i(k)$ describes how many times each outgoing link of agent $i$ appears in the DMST of all agents at iteration $k$, including agent $i$ itself. The bid vector $b_i(k)$ describes the bid placed by the agent $i$ at iteration $k$, that is the link $(i, e_i)$ with (higher) weight $w_i.e_i$.

I. Phase 1 (Link Selection Process): In this phase, each agent picks a redundant link as a candidate for deletion. The process requires each agent first to compute its DMST and then to identify its outgoing link which has been used the less among all the DMSTs.

The pseudo-code of DECS for the Phase 1, at the generic iteration $k$, is shown in Algorithm 1. As the first operation, agent $i$ computes its DMST [22] (line 5 of Algorithm 1) using the OLD procedure. For each agent $i$, the output of this procedure is the updated weight vector $w_i(k)$ which contains the weights of all direct links belonging to the DMST at iteration $k$, and the link counter vector $y_i(k)$ which contains, for each outgoing link of the DMST, the number of times each link is contained in all $n$ DMSTs. The values of this vector are obtained counting the requests sent to agent $i$ from all the other agents in $V$ (for further details on the implementation of DMST, please refer to [22]). In this way, if the direct link $(i,j)$ is selected only by the DMST of agent $i$ we obtain $y_{ij}(k) = 1$. Once obtained these two vectors, agent $i$ first computes the number of its outgoing links $c_i$ (line 6), obtained by the OLD procedure, then it finds outgoing links selected only by a number lower than $c_i$ of DMSTs in the network $(h_{ij}(k) = I(y_{ij}(k) < c_i))$, where the operator $I(\cdot)$ is equal to 1 when the argument is true. If there is at least one link which meets the aforementioned condition (line 8), agent $i$ places a bid on it. If there is more than one link which satisfies the condition, the one
Algorithm 2 - DECS - Phase 2 for agent $i$ at iteration $k$

SEND: $b_i(\tau)$ to $j \in N_i(t)$
RECEIVE: $b_j(\tau)$ from $j \in N_i(t)$

1: for $j = 1 : |N_i(t)|$ do
2: $b_{1j}(\tau) = \max \{b_{11}(\tau), b_{2j}(\tau)\}$
3: $z_i(\tau) = \arg\max\{b_{11}(\tau), b_{2j}(\tau)\}$
4: if $z_i(\tau) = 2$ then
5: $b_{22}(\tau) = b_{2j}(\tau)$
6: $e_i(k) = 0$
7: end if
8: end for

with the maximum weight is selected $(i, e_i)$ (line 9) and the correspondent weight is used as the bid for the agent $i$ at the current iteration $k$ (line 10). In the case that the agent has not the candidate link, the bidding process is skipped, and the agent’s bid remains set to zero ($b_{11} = 0$).

2) Phase 2 (Consensus for Deletion): In this phase, the redundant link with maximum weight is selected for deletion among all the candidates. This process requires all the agents to reach a consensus on the link to be deleted. A simple auction mechanisms based on the standard max-consensus protocol is used.

More specifically, the Phase 2 is iterated (where $\tau$ denote the iteration number) until all agents in $\mathcal{V}$ converge on the vector of bids. This vector will be used to determine the single link to be delete at iteration $k$. This procedure is shown in Algorithm 2. Since a communication link is associated to each sensing link, we can define $N_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}(t)\}$ as the communication neighborhood of agent $i$. At each iteration $\tau$ of phase 2 of the DECS algorithm, agent $i$ receives the bid vector $b_i(\tau)$ from each of its neighbors ($j \in N_i$). The consensus is performed on the first component of bid vector $b_i(\tau)$ in a way that agent $i$ updates $b_{11}(\tau)$ with the maximum value between its own bid and the bid of its neighbor agent $j$ (line 2). Then, the index related to the the maximum bid is stored in $z_i(\tau)$ (line 3). If $z_i(\tau)$ is equal to 2 this means that the maximum value is owned by agent $j$, thus the agent $i$ updates the second component of the bid vector $b_{22}(\tau)$ (line 5) and loses its candidate link $e_i = 0$ (line 6). We assume that ties may occur in this phase. This ties are resolved using a tie-breaking heuristic based on the agent identification number stored in the second component of $b_i(\tau)$. The selected link is deleted from the outgoing neighborhood of the winning agent. This process converges in a number of iterations equal to the diameter of the network [23]. Thus in the worst case scenario $n - 1$ iterations are required.

After this procedure the algorithm returns to the Phase 1, and new links to be eliminated are searched. The algorithm ends when the auction in Phase 2 converges on a bid equal to 0 ($b_i = [0, 0], \forall i \in \mathcal{V}$), this means that there are not more links to be deleted thus the $D'$ digraph is obtained.

IV. THEORETICAL ANALYSIS

In this section, a theoretical analysis of the properties and computational complexity of the proposed algorithm is provided.

A. Algorithm Properties

Regarding the algorithm properties, first it will be proven that each link deletion is safe with respect to the strong connectivity of the graph, successively it will be shown that the solution of the algorithm is ameliorative and finally it will be proven that the convergence is reached in a finite number of steps.

**Lemma 1:** Each link deletion carried out in the Phase 2 of the DECS algorithm is safe with respect to the strong connectivity property of the subgraph $D'$.

**Proof:** In order to prove the lemma let us point out that the link selection carried out in Algorithm 1 at line 7 avoids the disconnection of the graph in two clusters. This can be explained by the fact that if the link $(i, j)$ selected by the agent $i$ for the deletion has been chosen by a number of spanning trees $y_{ij}$ lower than the cardinality of its outgoing links $c_i$, there must always be an alternative path in the graph that allows node $i$ to reach node $j$, namely by exploiting at least one of the spanning trees computed by the agents belonging to its outgoing neighborhood. This can be proven by contradiction as follows. Let us assume the cardinality of the outgoing links of agent $i$ to be $c_i$, and let us assume the link $(i, j)$ selected by the agent $i$ for the deletion has been chosen by a number of spanning tree $y_{ij}$, with $y_{ij} < c_i$. Furthermore, let us assume the graph gets disconnected by the removal of this link. This implies that any agent $k$, with $k \neq j$ belonging to the outgoing neighbors of the agent $i$ can no longer reach the agent $j$. Therefore, the link $(i, j)$ were used also by all the agents $k$ with $k \neq j$ belonging to the outgoing neighbors of the agent $i$ to reach agent $j$. Thus it must be $y_{ij} = c_i$, which gives the absurd. ■

**Lemma 2:** Each link deletion is diminishing in terms of number of links and total weight of the strongly connected subgraph $D'$

**Proof:** The proof of this lemma comes from the fact that each weight is positive, thus each time a link is removed both the overall number of links and the total weight of the strongly connected subgraph $D'$ are diminished. ■

**Lemma 3:** The DECS algorithm converges towards an approximation of the CSS in at most $O(n \delta^+_{\text{max}})$ steps, where $\delta^+_{\text{max}} = \max_{e \in \mathcal{V}} (|\delta^+_e|)$ is the cardinality of the largest outgoing neighborhood.

**Proof:** In order to prove this lemma, let us recall that the proposed algorithm eliminates one link at each iteration. Now, by denoting with $\delta^+_{\text{max}}$ the cardinality of the largest outgoing neighborhood and by noticing that the solution with the lowest number of links is the Hamiltonian cycle, it follows that at most $n \delta^+_{\text{max}} - n$ links must be deleted. Therefore, the algorithm must be executed at most $O(n \delta^+_{\text{max}})$ steps. Furthermore, from Lemma 1 and Lemma 2
it follows that at each iteration the obtained subgraph $D'$ is strongly connected and it has a lower number of links and smaller total weight with respect to graph from which it is obtained. Thus, the algorithm converges towards an approximation of the CSS.

B. Algorithm Computational Complexity

Regarding the computational complexity, the worst case scenario for the fully and sparsely connected graph will be analyzed. The analysis is carried out for $n \geq 3$ as no deletion would be required otherwise.

1) Fully Connected Graph: In the case of a fully connect graph each agents has exactly $O(n)$ neighbors. In the worst case scenario, that is all the links have the same weight, the spanning tree routed at each agent will be given by the set of outgoing links of the agent itself. Therefore, the set of spanning trees involves $O(n^2)$ links. Since the proposed algorithm eliminates 1 link at each iteration, the worst case scenario is the one for which the algorithm converges towards a solution involving only $n$ links, i.e. a ring. Thus, the maximum number of links to be deleted is $O(n^2 - n)$, that is $O(n^2)$. Regarding the computational complexity of the proposed algorithm for a graph fully connected, at each iteration the Humblet algorithm takes $O(n^2)$ steps [22], while the max-consensus is carried out in $O(n)$ steps. Therefore the maximum number of steps required to converge towards a ring is $O(n)O(n^2)$, that is $O(n^3)$.

2) Sparsely Connected Graph: In the case of a sparsely connect graph each agents has $O(\log(n))$ neighbors. In the worst case scenario, that is all the links have the same weight, the spanning tree routed at each agent will be given by the set of outgoing links of the agent itself plus some additional links to reach each node. Therefore, the set of spanning trees involves at maximum $O(n \log(n))$ links. Since the proposed algorithm eliminates 1 link at each iteration, the worst case scenario is the one for which the algorithm converges towards a solution involving only $n$ links, i.e. a ring. Thus, the maximum number of links to be deleted is $O(n \log(n) - n)$, that is $O(n \log(n))$. Regarding the computational complexity of the proposed algorithm for a graph sparsely connected, at each iteration the Humblet algorithm takes $O(n \log(n))$ steps [22], while the max-consensus is carried out in $O(n)$ steps. Therefore the maximum number of steps required to converge towards a ring is $O(n \log(n))O(n \log(n))$, that is $O(n^2 \log^2(n))$.

V. SIMULATION RESULTS

A Monte Carlo simulation campaign has been performed to validate the effectiveness of the proposed DECS. For each trial, a strongly connected digraph $D$ by $n$ nodes, and $m$ links was considered. For the sake of simplicity, the initial strongly connected digraph $D$ was obtained by including a Hamiltonian cycle, of $n$ links, plus $m - n$ additional links randomly placed. Note that, as explained in Section III, the optimal solution might not be unique, but rather there could be a set of optimal solutions, namely the Pareto front. Furthermore, the identification of a CSS is a NP-hard problem. For these reasons, we analyze the algorithm performance in terms of number of links $L$ and total weight $W$ separately. Regarding the first index $L$, a lower bound is given by the Hamiltonian cycle which has exactly $n$ links, i.e., $\text{LB}_{\text{links}} = n$. Note that, in our simulations this lower-bound is the solution with the minimum number of links as we assume a Hamiltonian cycle to be always embedded in the initial graph. Regarding the second index $W$, a lower-bound can be computed as follows:

i) we find all the $n$ rooted Directed Minimums Spanning Trees $T_i \forall i \in V$ and for each $T_i$;

ii) we add, to the set of links $E^*_i$, the ingoing link $(j^*, i)$ of node $i$ with the minimum weight, $E^*_i = E^*_i \cup (j^*, i)$;

iii) we select among these updated DMSTs the the one with the minimum total weight as the lower bound value i.e., \( \text{LB}_{\text{weight}} = \min_i \sum_{(i,j)} E^*_i w_{ij} \).

Figure 3 shows the simulation results. Simulations were carried out by ranging the number of robots from 5 to 30 has been considered. For each configuration 20 trials were run and average valued computed.

On Figure 3a the y-axis denotes the average number of links while the x-axis is the number of agents. The red line represents the lower-bound $\text{LB}_{\text{links}}$, that is the the Hamiltonian cycle with $n$ links. The green (dashed) line represents two times the lower-bound, namely $2n$. Finally, the blue line represents the average number of links obtained with the proposed DECS, while the blue dots represent the cardinality of the actual set of links of each solution obtained for each configuration (a fixed number of agents). It can be noticed that the solution provided by the algorithm is always less than two times the lower bound. Indeed, this seems a promising result considering the distributed nature of the algorithm.

On Figure 3b the y-axis denotes the total weight while the x-axis is the number of agent is given. The red line represents the lower-bound $\text{LB}_{\text{weight}}$ obtained according to the procedure explained above. The blue line represents the average total weight obtained with the proposed DECS algorithm. Also in this case the algorithm solution can be considered a good approximation respect to the lower-bound of the minimal total weight.

To analyze the scalability of the algorithm with respect to the number of links, we ranged the additional links $m - n$ according to the Gilbert random graph model, i.e., an edge is added with probability $p$. In this way, for $p = 0$ a Hamiltonian cycle is obtained while for $p = 1$ a fully connected graph is achieved. On Figure 3c the y-axis denotes the number of total links $L$ while the x-axis is the connectivity probability $p$ for the additional $m - n$ links. In Figure 3c we show the results of the simulations of 5 configuration of networks ($n = 5, 10, 15, 20, 25$) where the number of links of the solution of DECS is evaluated for different connection probability $p$. Note that the DECS algorithm performs well for graph with small $p$, due to the small number of choices. However, increasing the number of
possible choices, the DECS has a constant behavior, always under two time the optimal value \((2 - \text{LB}_{\text{links}} = 2n)\), from \(p = 0.5\) to a fully connected graph \(p = 1\).

VI. CONCLUSION

In this work a decentralized algorithm for the connectivity control of digraphs has been proposed. A theoretical analysis along with a numerical simulation campaign has been carried out to show the effectiveness of the proposed approach. This represents the first step towards the development of a distributed framework for the connectivity maintenance of heterogeneous robotic networks with limited Field Of View. Future work, will be mainly focused on providing a theoretical characterization on the bound of the CSS approximation. Furthermore, the design of a proper control law for the connectivity control framework will be also investigated.

REFERENCES


