SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 4, NO. 3, DECEMBER 2021
DOI: 10.35377/saucis.04.03. 901776

GraParT: A MATLAB Toolbox for Partitioning Directed Graphs

Onur CİHAN
Marmara University, Faculty of Engineering, Department of Electrical and Electronics Engineering;
onur.cihan@marmara.edu.tr

Received 23 March 2021; Accepted 7 September 2021; Published online 31 December 2021

Abstract

Consensus algorithms are increasingly used in multi-agent systems due to their advantages in various applications. Recent results on consensus algorithms show that the number of groups formed in a network of agents utilizing consensus-based algorithms can be computed once its primary and secondary layer subgraphs are determined. In this study, we present GraParT -Graph Partitioning Toolbox- that can be used to partition directed graphs by determining its primary and secondary layer subgraphs and the vertices therein. The toolbox helps the user to build, modify, analyze and illustrate directed graphs in terms of the grouping behavior of the consensus algorithms with its user-friendly interface. GraParT is an open-source software that is available free of charge for academic and non-commercial use.

Keywords: directed graphs, consensus algorithms, group consensus, MATLAB toolbox

1. Introduction

The last two decades have seen the rapid development of consensus algorithms due to their use in practical problems consisting of multiple agents [1-6]. These algorithms find applications in the fields of robotics [1, 2], computer networks [3], distributed optimization [4], and social networks [5]. Most of the work consider the consensus problem as agreement on a single state which requires the underlying graph of the network to have a spanning tree [7, 8]. If there is no spanning tree in the graph, consensus on a single state is not possible and multiple groups will be formed in the network [8].

Recently, the number of groups that will be formed in a multi agent network utilizing a consensus based algorithm was investigated for networks with first [9], second [10] and third order agent dynamics [11]. While the stability conditions are different for these networks, the grouping behaviors are the same. The concepts of primary layer subgraphs (PLS) and secondary layer subgraphs (SLS), which were first introduced in [9], have been instrumental in our understanding of the grouping behavior of a multi-agent system. Once these subgraphs are determined, the topology designer can merge or divide the groups by adding new edges to the graph [12].

Hespanha proposed a MATLAB algorithm to partition undirected graphs which can be used to solve the l-bounded Graph Partitioning (l-GP) problem [13]. In this problem, the vertices of the given undirected graph are partitioned into k disjoint subsets where the sum of the edges connecting the vertices in different subsets is minimized. Çatalyürek and Aykanat built a hypergraph partitioning tool called PaToH that can be utilized to solve various combinatorial scientific computing problems including VLSI layout design and dynamic load balancing for parallel processing [14]. The objective of such partitioning is to divide the given graph into two (or more) subgraphs with equal (or nearly equal) number of vertices such that the cost function defined on the hyperedges connecting vertices in different subgraphs is minimized. Furthermore, depending on the problem, another objective may be to keep the sum of hyperedges connecting the vertices in the same subgraphs as close as possible (load balancing problem). For a detailed survey on graph partitioning, we refer the reader to [15] and the references therein. While some research has been carried out on developing graph partitioning tools for solving different graph-related problems, to the best of our knowledge, this is the first study that provides a tool for partitioning directed graphs to determine the groups and their members in a network of agents utilizing a consensus-based algorithm. Due to the applicability of the consensus algorithms in important real-world problems

(such as formation control of multi-robot systems, analysis of social network analysis, etc.), a toolbox for partitioning directed graphs may help the engineers in designing network topologies.

In this paper, our objective is to provide a tool for determining these groups and the vertices in each group by expoiting the graph structure. This tool enables the network topology designer to understand the grouping behavior of the multi-agent system without running a consensus based algorithm. More specifically, the topology designer can add new links or remove existing ones to form a new network or modify an existing one to obtain desired grouping when a consensus based algorithm is used. For instance, in the formation control problem of a multi-robot system, this tool can be used to design a network topology where the robots in the same group achive a desired formation. To this end, we utilize the concepts of PLS and SLS; and implement the detection algorithms whose pseudocode were given in [8]. Moreover, we propose the notion of *reduced graphs* where each group is represented by a vertex and the interaction between the agents in two different groups are represented by an edge in the reduced graph. Finally, we propose an algorithm for obtaining the reduced graph of an arbitrary directed graph and discuss its time and space complexity. With this novel definition, one can analyze complex directed graphs in terms of the grouping behaviors of the agents that are utilizing a consensus based algorithm. To the best of authors' knowledge, this is the first study to provide such a graph theoretic concept and an algorithm its detection.

We organize the remaining parts of this paper as follows. We review graph theory concepts that are used throughout the paper and give the mathematical formulation of the conventional continuous-time and discrete-time consensus algorithms with linear agent dynamics in Section 2. In Section 3, we present the workflow of the toolbox; and demonstrate a short tutorial on how to use the toolbox, interpret the outputs and visualize the partitioned graph. Finally, Section 4 concludes the paper.

2. Graph Theory Preliminaries and Dynamical Model of Consensus

2.1 Graph Theory Preliminaries

Communication network of a multi-agent system is modeled by a directed graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ where $\boldsymbol{V} = \{\boldsymbol{v_1}, \dots, \boldsymbol{v_n}\}$ is the finite set of vertices and $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ is the set of edges representing the information flow between agents. We say that there exists a directed edge form $\boldsymbol{v_i}$ to $\boldsymbol{v_j}$ if $(\boldsymbol{v_i}, \boldsymbol{v_j}) \in \boldsymbol{E}$ and $\boldsymbol{v_i}$ is called a neighbor of $\boldsymbol{v_j}$. The graph $\boldsymbol{G}$ is said to consist of a spanning tree if there exists a vertex $\boldsymbol{v_r}$ such that all vertices in $\boldsymbol{G}$ receive information from $\boldsymbol{v_r}$ directly or indirectly (in more than 1 step). Here, $\boldsymbol{v_r}$ is called a root of the graph $\boldsymbol{G}$. When there is no spanning tree in a directed graph, it can be partitioned into its PLS and SLS whose definitions are given as follows.

Definition 1. (Primary and secondary layer subgraphs)

Let the multi-agent system be modeled by a directed graph $G = (V, E)$. Then G can be partitioned into $l_p + l_s$ subgraphs such that

i) the vertices in l_p subgraphs (each consisting of the largest possible spanning tree) are not connected to the rest of the graph, and

ii) the vertices in l_s subgraphs (each consisting of a spanning tree) are connected to the rest of the network through a single vertex which is a root of the subgraph and the only vertex to receive information from other subgraphs. Furthermore, this root vertex has at least two neighbors in two other subgraphs.

The subgraphs defined in items (i) and (ii) are called the PLSs and SLSs subgraphs of G and denoted by $G_{p,i}$ $(i = 1, \dots, l_p)$ and $G_{s,j}$ $(j = 1, \dots, l_s)$ respectively. Note that this partitioning is unique for a given directed graph.

2.2 Dynamical Model of Consensus

Consider a network of n agents evolving over a directed graph $G = (V, E)$. The conventional distributed consensus algorithm with continuous time first-order agent dynamics can be expressed as

$$\dot{x}_i(t) = \sum_{j \in N_i} a_{ij} \left(x_j(t) - x_i(t) \right), \qquad i = 1, \dots, n \tag{1}$$

where $x_i(t) \in R^m$ is the state of agent i at time t, N_i is the set consisting of the neighboring vertices of agent i and a_{ij} is the (i, j)-th element of the adjacency matrix A defined as $a_{ij} > 0$ if $(v_j, v_i) \in E$ and $a_{ij} = 0$ otherwise. In matrix form, the system represented by Equation 1 can be expressed as

$$\dot{x}(t) = -(L \otimes I_m) x(t) \tag{2}$$

where I_m is the $m \times m$ identity matrix, $\otimes$ is the Kronecker product operator and $L = [l_{ij}]$ is the Laplacian matrix defined by

$$l_{ij} = \begin{cases} \sum_{\substack{k=1 \\ k \neq i}}^{n} a_{ik}, & \text{if } i = j \\ -a_{ij}, & \text{otherwise} \end{cases} . \tag{3}$$

In discrete-time, the distributed algorithm with first-order agent dynamics is given as follows

$$x_i(k+1) = w_{ii} x_i(k) + \sum_{j \in N_i} w_{ij} x_j(k), \qquad i = 1, \dots, n \tag{4}$$

where $x_i(k) \in R^m$ is the state of agent i at time step k, and $w_{ij} \geq 0$ is the weighting coefficient corresponding to the information exhange between agents j and i. The following assumption is necessary for the stability of the system represented by Equation 4.

Assumption 1. The following conditions hold for the weighting coefficients w_{ij}

$$i) \quad w_{ij} \begin{cases} > 0, & \text{if } (v_j, v_i) \in E \text{ or } i = j \\ = 0, & \text{otherwise.} \end{cases} \tag{5}$$

$$ii) \quad \sum_{j=1}^{n} w_{ij} = 1, \text{ for all } i \in \{1, \dots, n\} \tag{6}$$

Assumption 1(i) ensures that the weighting coefficients are always non-negative, and equal to 0 if there is no information flow between agent j to agent i. Assumption 1(ii) guarantees that the weighting coefficients sum up to 1 for all i.

The system represented by Equation 4 can be expressed in matrix form as

$$x(k+1) = (W \otimes I_m) x(k) \tag{7}$$

where $x(k) = [x_1(k)^T, \dots, x_n(k)^T]^T$ and $W = [w_{ij}]$ is the weighting matrix of the system.

Given a multi-agent system with a directed graph that is not consisting of a spanning tree, the agents can not achieve consensus on a single equilibrium state. In such a case, the agents converge to different vectors and multiple groups are formed. The definition of group consensus is introduced as follows.

Definition 2. (Group consensus)

We say that the multi-agent system with the underlying graph G achieves group consensus if there are c nonempty sets S_l $(l = 1, \dots, c)$ such that

$$\bigcup_{l=1}^{c} S_l = V, \quad S_l \cap S_q = \emptyset, \quad \text{for } l \neq q \text{ and } l, q = 1, \dots, c \tag{8}$$

and for the set S_l we have

$$\lim_{k\to\infty} \left\| x_i(k) - x_j(k) \right\| = 0, \quad \forall v_i, v_j \in S_l \tag{9}$$

for arbitrary initial conditions $x_i(0) \in R^m$ and arbitrary choice of averaging coefficients w_{ij} satisfying Assumption 1 (or equivalently, arbitrary choice of a_{ij}).

Note that for a multi-agent network with dynamics given in Equation 1 (or equivalently Equation 4 for discrete-time networks), the number of groups is related to the structure of the network [9-12, 16]. The following lemma states relationship between the number of groups and the graph partitioning based on Definition 1.

Lemma 1. (Number of groups)

Consider a multi-agent system with the underlying graph G where agents utilize the consensus algorithm given by Equation 1 (or equivalently Equation 4 for discrete-time networks). The multi-agent system forms $c = l_p + l_s$ groups where l_p and l_s are the number of PLSs and SLSs of G, respectively [9].

From Lemma 1, one can conclude that the number of groups can be determined from the numbers of PLS and SLS of the underlying network. It is shown in [9] that the agents' states in a particular subgraph converge to the same state. Furthermore, the states of the agents in the SLSs asymptotically converge to a convex combination of those of the agents in the PLSs. The main motivation of this study is to provide a useful tool for determining these groups and their members for a given directed graph. To better illustrate the groups in the network the following definition is introduced.

Definition 3. (Reduced graph)

Let $G = (V, E)$ be a directed graph consisting of l_p PLSs and l_s SLSs. Let $G_i = (V_i, E_i)$ denote the PLS for $i = 1, \dots, l_p$ and the SLS for $i = l_p + 1, \dots, l_p + l_s$. Then the graph $\bar{G} = (\bar{V}, \bar{E})$ is called the reduced graph of G where $\bar{V} = \{\bar{v}_1, \dots, \bar{v}_{l_p+l_s}\}$ is the vertex set and $\bar{E}$ is the edge set of the reduced graph whose elements are defined as

$$(\bar{v}_i, \bar{v}_j) \begin{cases} \in \bar{E} & if\ \exists v_a \in V_i, v_b \in V_j \text{ such that } (v_a, v_b) \in E \\ \notin \bar{E} & \text{otherwise} \end{cases}. \tag{10}$$

The adjacency matrix of the reduced graph is a block matrix of the form

$$\bar{A} = \begin{bmatrix} 0_{l_p \times l_p} & 0_{l_p \times l_s} \\ \bar{A}_{sp} & \bar{A}_s \end{bmatrix} \tag{11}$$

where $\bar{A}_s$ is related with the communication between secondary layer subgraphs and $\bar{A}_{sp}$ refers to the one-way communication between the PLS and SLS. Here $0_{m\times n}$ denotes the zero matrix of dimension $m \times n$.

In [9], two algoritms were introduced to determine the PLS and SLS of a given directed graph. Once these subgraphs are determined, the following algorithm can be used to determine the reduced graph $\bar{G}$.

Algorithm 1 Reduced graph extraction algorithm

```
1  procedure Ḡ = ReducedGraph(G, G_1, ..., G_{l_p+l_s}, l_p, l_s)
2  V̄ ← {v̄_1, ..., v̄_{l_p+l_s}}
3
4  Ē ← ∅
5  for i ← 1, l_p do
6     for j ← 1, l_s do
7        for all v_a ∈ V_i do
8           for v_b ∈ V_{l_p+j} do
9              if (v_a, v_b) ∈ E then
```

```
                    E̅ ← E̅ ∪ (v̅_i, v̅_{l_p+j})
                end if
            end for
        end for
    end for
end for
for i ← 1, l_s do
    for j ← 1, l_s do
        for all v_a ∈ V_{l_p+i} do
            for v_b ∈ V_{l_p+j} do
                if (v_a, v_b) ∈ E then
                    E̅ ← E̅ ∪ (v̅_{l_p+i}, v̅_{l_p+j})
                end if
            end for
        end for
    end for
end for
G̅ ← (V̅, E̅)
return G̅
end procedure
```

The algorithm initially generates a set consisting of $\boldsymbol{l_p+l_s}$ vertices with an empty set of edges (time complexity $\boldsymbol{O(n)}$, space complexity $\boldsymbol{O(n)}$). For each agent in the primary layer subgraphs, the algorithm checks whether there is a directed path to an agent in the secondary layer subgraphs (time complexity $\boldsymbol{O(n^2)}$) . If there exists such a path, a link is added to the edge set of the reduced graph (space complexity $\boldsymbol{O(n^2)}$). The same procedure is repeated for the links between the agents in the secondary layer subgraphs (time complexity $\boldsymbol{O(n^2)}$ and space complexity $\boldsymbol{O(n^2)}$). Consequently, the overall time complexity and space complexity of the reduced graph extraction algorithm is $\boldsymbol{O(n^2)}$. On the other hand, this algorithm requires the PLS and SLS of the graph to be determined prior to its execution. Since the time complexity of PLS and SLS detection algorithms are $\boldsymbol{O(n^4)}$ and $\boldsymbol{O(n^6)}$ (see [8]), we conclude that the time complexity of detecting the reduced graph of an arbitrary directed graph is $\boldsymbol{O(n^6)}$. Note from Definition 1 that the vertices in a PLS are not connected to the vertices in other PLS and SLS. Therefore $\boldsymbol{l_p}$ vertices in the reduced graph $\boldsymbol{\bar{G} = (\bar{V}, \bar{E})}$ do not receive information from other vertices. We would like to note also that the concept of a *reduced graph* is novel and therefore there does not exist any other algorithm in the literature which can be used to determine the reduced graph of an arbitrary directed graph.

3. GraParT: Graph Partitioning Toolbox

In this section, we present GraParT: a MATLAB toolbox for partitioning directed graphs.

3.1 Installation

GraParT can be downloaded from http://www.onurcihan.com/GraParT.html and requires R2018b or a newer release of MATLAB to work properly. It is compatible with Windows, macOS and Linux operating systems.

3.2 The Workflow

GraParT allows the users to input the directed edges of a graph and computes the adjacency matrix $\boldsymbol{A}$, the Laplacian matrix $\boldsymbol{L}$, the weighting matrix $\boldsymbol{W}$ (assuming equal weighting is used for the information coming from different agents), the partitions of the graph (namely, the PLS and SLS and the vertices

therein) and the reduced graph. Furthermore, evolution of the states of the agents utilizing Equation 1 is given as a visual output. The workflow of GraParT is shown in Figure 1.

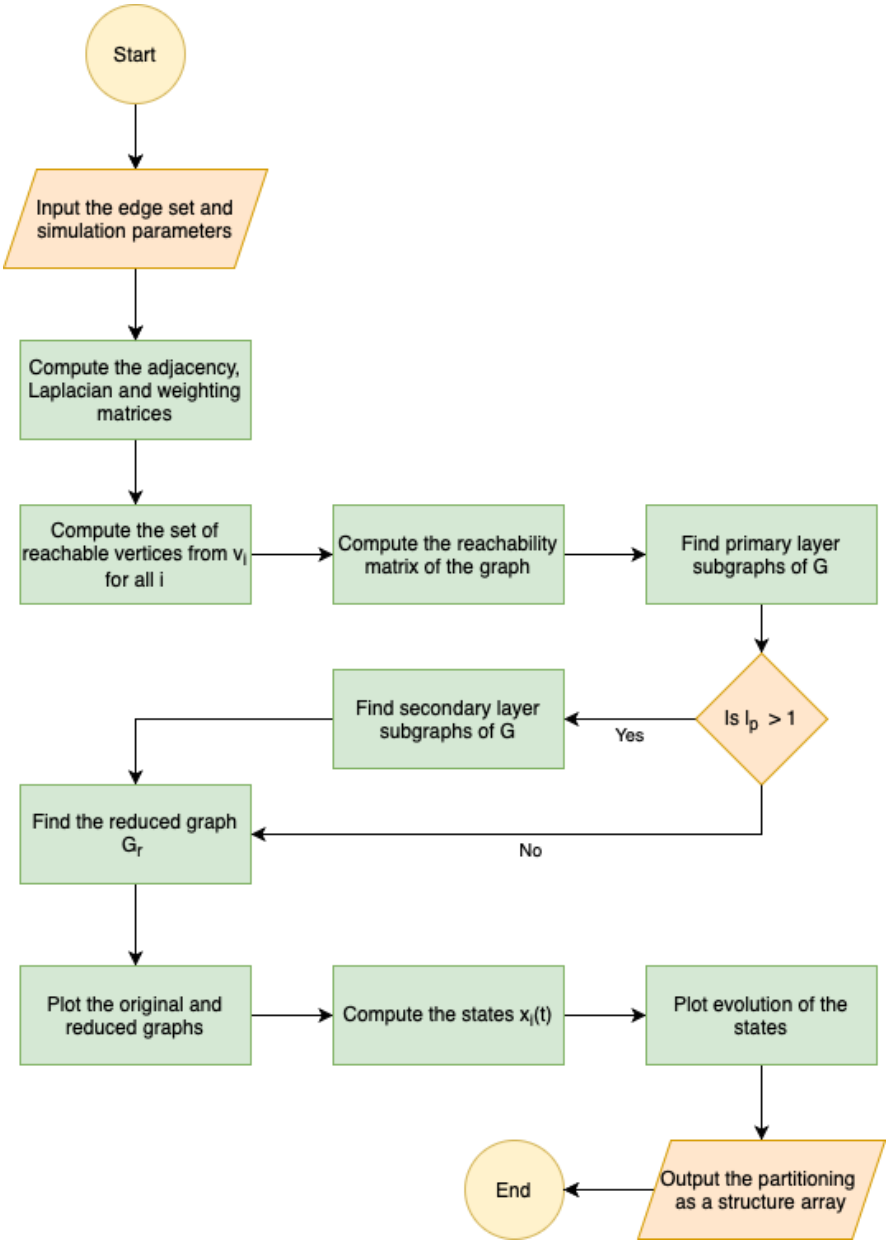

Figure 1 Overall workflow of GraParT

Note from Definition 1 that if a graph contains a single PLS, then all vertices of the graph is a member of the PLS and consequently the graph does not contain any secondary layer subgraphs. The PLS and SLS are determined by using the algorithms proposed in [9] and the reduced graph is determined by Algorithm 1.

3.3 The Graphical User Interface

GraParT has a user-friendly graphical user interface as illustrated in Figure 2.

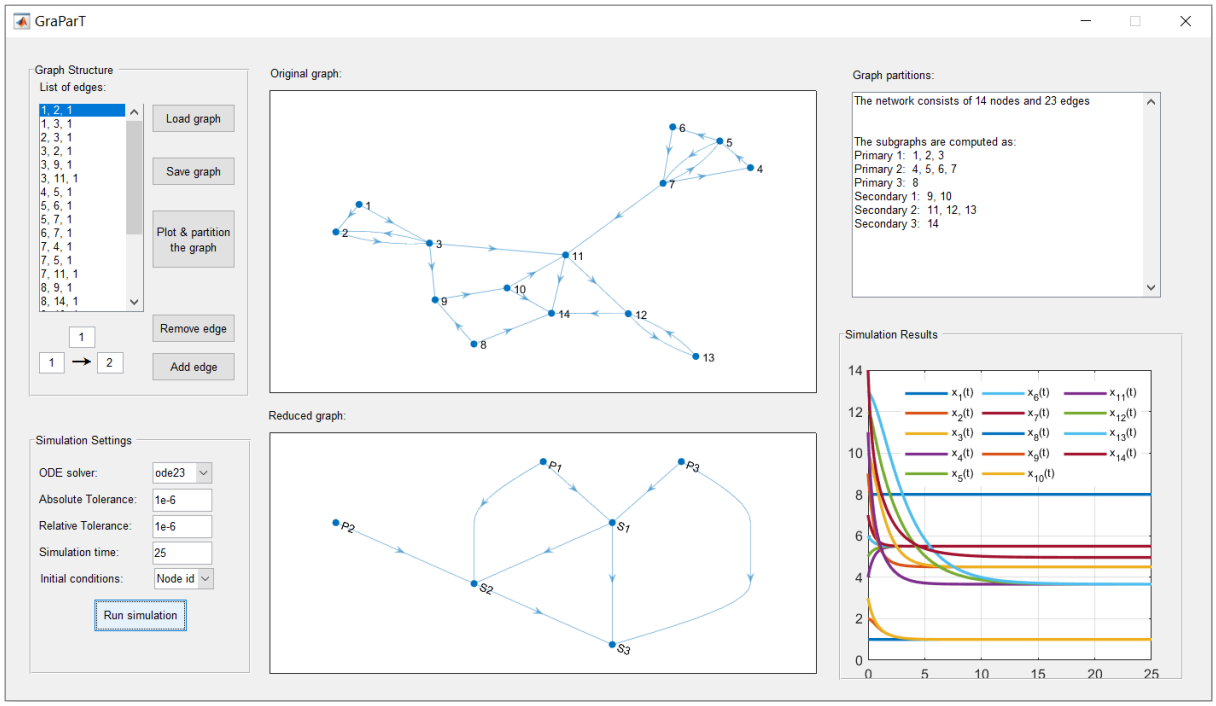

Figure 2 The Graphical User Interface of GraParT

The user enters the directed edges of the graph from the left-hand side of the user interface and the graphs can be saved as text files and loaded for future use. Once the user clicks the "Plot & partition the graph" button, GraParT computes the adjacency matrix, the Laplacian matrix and the weighting matrix; and partitions the graph into its PLS and SLS. The subgraphs and vertices are given as analysis results at the right-hand side of the interface. Furthermore, the reduced graph is computed and depicted together with the original graph.

The ordinary differential equation (ODE) solver that will be used to solve Equation 1, absolute and relative tolerances in the calculations, the maximum time for simulation (in seconds) and the initial states of the agents are simulation parameters to be entered by the user. The evolutions of the states are depicted as functions of time on the bottom-right side of the graphical interface and can be used to verify that the number of groups is equal to the number of subgraphs determined by GraParT.

In most multi-agent systems, the agents are mobile and as a result of this, the network connectivity may be time-varying. A new communication link may be built when two agents come closer, and an existing link may be broken when they move away from each other. In order to understand the grouping behavior of the network in such cases, we provide the following examples.

Example 1.

Consider a network of 14 agents whose edge set is given by

$$E_1 = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_3, v_2), (v_3, v_9), (v_3, v_{11}), (v_4, v_5), (v_5, v_6), (v_5, v_7), (v_6, v_7), (v_7, v_4), (v_7, v_5), (v_7, v_{11}), (v_8, v_9), (v_8, v_{14}), (v_9, v_{10}), (v_{10}, v_{11}), (v_{10}, v_{14}), (v_{11}, v_{12}), (v_{11}, v_{14}), (v_{12}, v_{13}), (v_{12}, v_{14}), (v_{13}, v_{12})\}. \quad (12)$$

Since there is no spanning tree in the graph $\boldsymbol{G_1} = (\boldsymbol{V_1}, \boldsymbol{E_1})$, multiple groups will be formed in the network. Once the directed edges are entered as inputs to GraParT, Figure 2 illustrates the network graph, the reduced graph; the PLS and SLS of $\boldsymbol{G_1}$, and evolutions of the states of the agents as functions of time. As can be seen from Figure 1, there are 3 PLS and 3 SLS in the network and the simulation

results verify that there are **6** consensus equilibria (groups of agents with same final values) in the network.

Example 2.

In order to understand the effect of adding edges to a graph on the grouping behavior of a multi-agent network, suppose that a new link between agents 8 and 13 is created. The edge set of the new graph can be written as

$$E_2 = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_3, v_2), (v_3, v_9), (v_3, v_{11}), (v_4, v_5), (v_5, v_6), (v_5, v_7), (v_6, v_7), (v_7, v_4), (v_7, v_5), (v_7, v_{11}), (v_8, v_9), (v_8, v_{14}), (v_9, v_{10}), (v_{10}, v_{11}), (v_{10}, v_{14}), (v_{11}, v_{12}), (v_{11}, v_{14}), (v_{12}, v_{13}), (v_{12}, v_{14}), (v_{13}, v_{12}), (v_8, v_{13})\}. \quad (13)$$

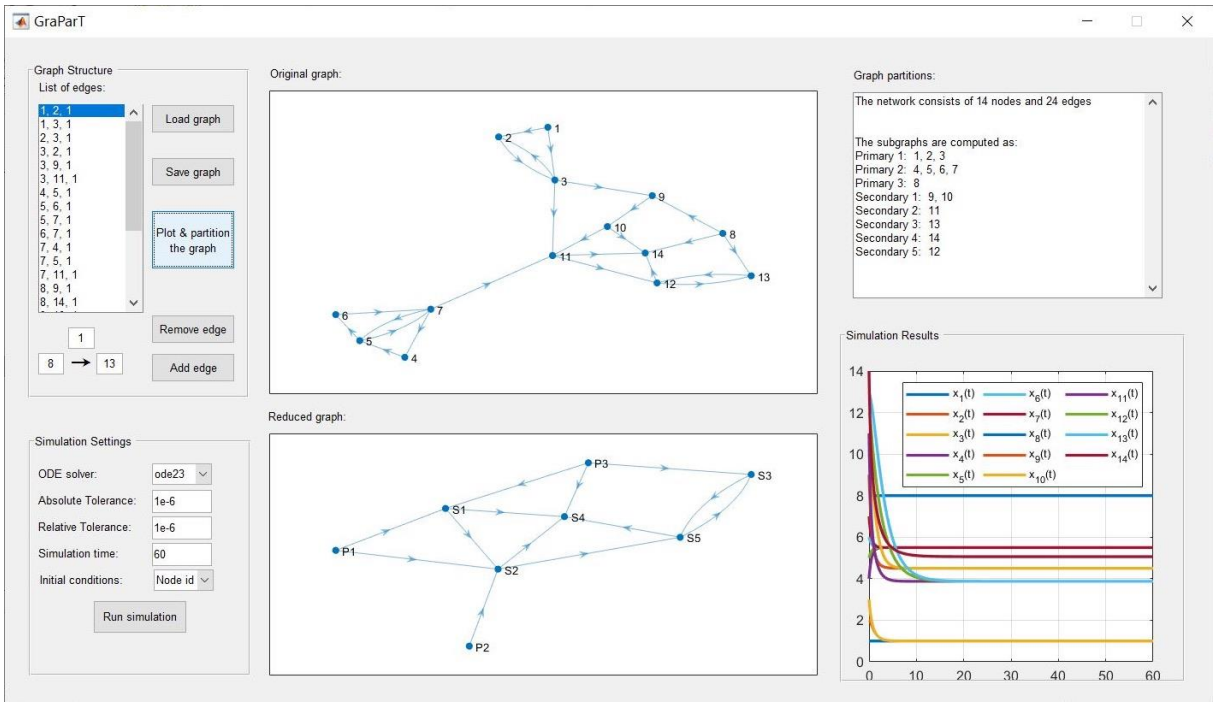

Figure 3 Partitioning of the network and the simulation results for Example 2

Figure 3 illustrates the network graph, the reduced graph; the PLS and SLS of $\boldsymbol{G_2}$, and evolutions of the states of the agents as functions of time. As can be seen from Figure 3, there are 2 PLS and 3 SLS in the network. In particular, the creation of the link between agents 8 and 13 resulted in the dissolution of the a secondary layer subgraph and formation of new secondary layer subgraphs in the modified graph. The total number of groups in the modified network is 8.

Example 3.

Reconsider the network under investigation in Example 2. Suppose that the link between agents 11 and 12 is removed. The edge set of the new graph can be written as

$$E_3 = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_3, v_2), (v_3, v_9), (v_3, v_{11}), (v_4, v_5), (v_5, v_6), (v_5, v_7), (v_6, v_7), (v_7, v_4), (v_7, v_5), (v_7, v_{11}), (v_8, v_9), (v_8, v_{14}), (v_9, v_{10}), (v_{10}, v_{11}), (v_{10}, v_{14}), (v_{11}, v_{14}), (v_{12}, v_{13}), (v_{12}, v_{14}), (v_{13}, v_{12}), (v_8, v_{13})\}. \quad (14)$$

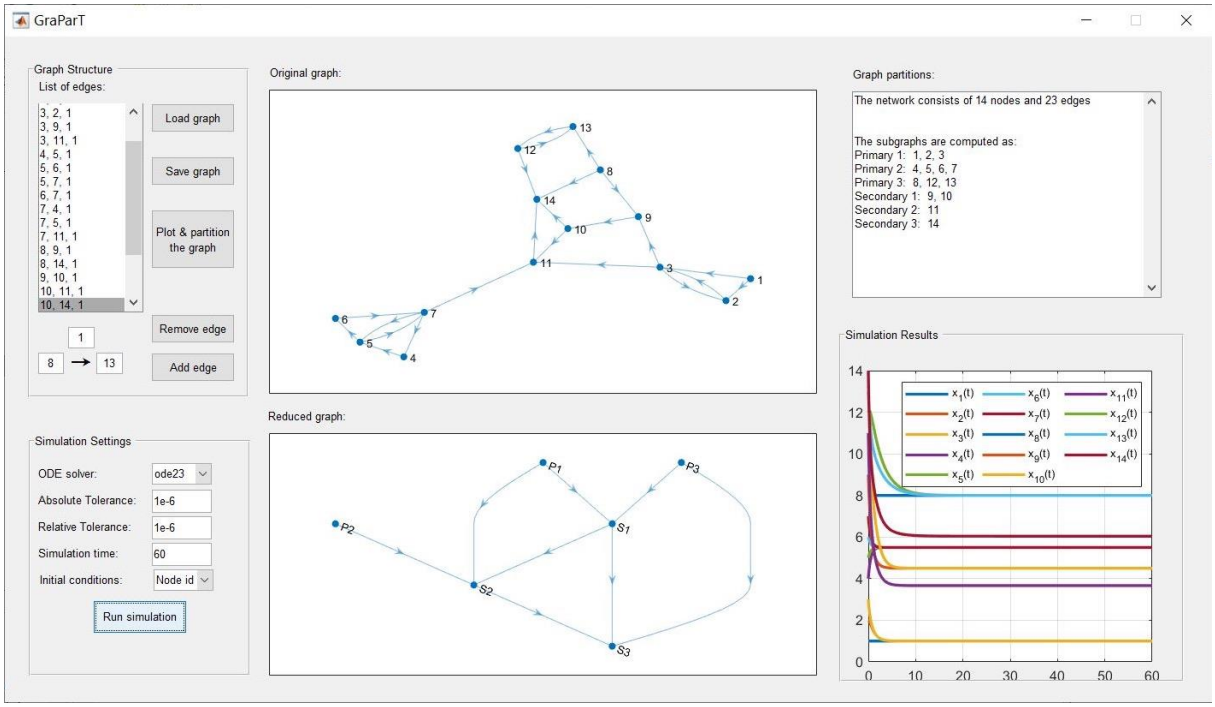

Figure 4 Partitioning of the network and the simulation results for Example 3

Figure 4 illustrates the network graph, the reduced graph; the PLS and SLS of $\boldsymbol{G_3}$, and evolutions of the states of the agents as functions of time. As can be seen from the figure, there are 3 PLS and 3 SLS in the network. In particular, the removal of the link between agents 11 and 12 resulted in unification of two SLS and a primary layer subgraph into a large primary layer subgraph (with the member agents 8, 12 and 13). The total number of groups in the modified network is 6.

Examples 1-3 show that modification of a directed graph by adding a new link or removing an existing link may result in an increase or a decrease of the number of groups formed in the network. By using the proposed partitioning tool, the network topology designer can create a directed multi-agent network with the desired grouping behavior.

4. Conclusion

In this paper, we provide a MATLAB toolbox for partitioning directed graphs into its PLS and SLS. The states of agents in these subgraphs asymtoyically converge to the same values when they utilize a conventional continuous-time or discrete-time consensus algorithm. The toolbox enables the users to modify the graph and analyze the effect of these changes in the graph structure to the number of groups formed in the multi-agent network. We hope this toolbox will enrich the understanding of the grouping behavior of the multi-agent systems and will be a reference tool for network topology designers.

References

[1] R. Aragues, J. Cortes, and C. Sagues, “Distributed consensus on robot networks for dynamically merging feature-based maps, ” IEEE Trans. Robot., vol. 28, no. 4, pp. 840–854, 2012.

[2] M. Mirzaei, H. Atrianfar, N. Mehdipour, and F. Abdollahi, “Asynchronous consensus of continuous-time lagrangian systems with switching topology and non-uniform time delay, ” Rob. Auton. Syst., vol. 83, pp. 106–114, 2016.

[3] N. Amelina, A. Fradkov, Y. Jiang, and D. J. Vergados, “Approximate consensus in stochastic

networks with application to load balancing," IEEE Trans. Inform. Theory, vol. 61, no. 4, pp. 1739–1752, 2015.

[4] O. Cihan, "Distributed Solution of Road Lighting Problem Over Multi-Agent Networks, " Sakarya University Journal of Computer and Information Sciences, vol. 3, no. 2, pp. 89–98, 2020.

[5] R. Hegselmann and U. Krause, "Opinion dynamics and bounded confidence: Models, analysis and simulation, " J. Artif. Soc. Soc. Simul., vol. 5, no. 3, pp. 1–33, 2002.

[6] O. Cihan, "Rapid solution of linear equations with distributed algorithms over networks, " IFAC-PapersOnLine, vol. 52, no. 25, pp. 467-471, 2019.

[7] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays, " IEEE Trans. Automat. Control, vol. 49, no. 9, pp. 1520–1533, 2004.

[8] W. Ren and R. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies, " IEEE Trans. Automat. Control, vol. 50, no. 5, pp. 655–661, 2005.

[9] Ö. F. Erkan, O. Cihan, and M. Akar, "Analysis of distributed consensus protocols with multi-equilibria under time-delays, " J. Franklin Inst., vol. 355, no. 1, pp. 332–360, 2018.

[10] O. Cihan and M. Akar, "Multi-consensus of second-order agents in discrete-time directed networks, " Int. J. Syst. Sci., vol. 51, no. 10, pp. 1847-1861, 2020.

[11] O. Cihan and M. Akar, "Necessary and Sufficient Conditions for Group Consensus of Agents With Third-Order Dynamics in Directed Networks, " J. Dyn. Syst. Meas. Control, vol. 142, no. 4, pp. 041003, 2020.

[12] O. Cihan, "Topology design for group consensus in directed multi-agent systems, " Kybernetika, vol. 56, no. 3, pp. 578–597, 2020.

[13] J. Hespanha, "An efficient MATLAB Algorithm for Graph Partitioning, " Technical Report, University of California, Oct. 2004.

[14] Ü. Çatalyürek and C. Aykanat C, "PaToH (Partitioning Tool for Hypergraphs), " In: Padua D. (eds) Encyclopedia of Parallel Computing. Springer, Boston, MA, 2011.

[15] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, "Recent Advances in Graph Partitioning, " In Algorithm Engineering, pp. 117–158, Springer International Publishing, 2016.

[16] Ü. Develer and M. Akar, "Cluster consensus in first and second-order continuous-time networks with input and communication delays," International Journal of Control, vol. 9 no. 4, pp. 961-976, 2021.