Controlling Best Response Dynamics for Network Games

MohammadAmin Fazli, Abbas Maazallahi, Jafar Habibi, and Moslem Habibi

Abstract—Controlling networked dynamical systems is a challenging endeavor, specifically keeping in mind the fact that in many scenarios the actors engaged in the dynamism behave selfishly, only taking into account their own individual utility. This setting has been widely studied in the field of game theory. One way we can control system dynamics is through the use of control parameters that are at our disposal, but finding optimal values for these parameters is complex and time consuming. In this paper we use the relation between network structural properties and control parameters to create a mathematical model that speeds up the calculation of the aforementioned values. For this, we use learning methods to find optimal values that can control the system dynamics based on the correlation between structurally similar networks.

Index Terms—Networks, best response dynamics, game theory, control problems, machine learning, heuristics

1 Introduction

Controlling dynamical systems, that is being able to guide such systems towards states with desirable outcomes, is a challenging and complex endeavor. The importance of these systems and the noticeable roles they play in various scientific fields make this issue one which has seen ever growing contributions from the scientific community. Specially, the growth of networked dynamical systems and their inherent complex internal interactions has attracted the interest of many researchers [1], [2], [3], [4]. In networked dynamical systems, a connected underlying structure of interconnecting links is present and thus an interdependence between the behaviors of individual actors who are involved in the system can be seen [5]. In such systems, the behavior for each actor depends not only on his own actions, but also on the actions chosen by other actors.

In this paper, we consider the problem of controlling networked dynamical systems that have individual, intelligent and selfish actors. This is the case in many sociological and economical contexts. In such systems, each actor usually undertakes actions keeping in mind his own benefit. Controlling the state of the system under such an assumption without forcing or guiding the individual actors in some way is not an easy task. Game theory is a mathematical way to study interactions between selfish agents in such contexts [6], [7]. It has many applications in such diverse disciplines as economics, political sciences, physics and biology [8].

In game theory, we model interactions between individual agents as a game where each agent is a so-called player. Each player has a utility function which he tries to maximize by deciding on an action to take. The basic assumption in classical game theory is that each player acts rationally whilst at the same time assuming that other players also behave similarly [9]. One of the subbranches of game theory, commonly referred to as network games, is devoted to the study of different aspects of networked systems such as topology formation, routing and congestion control [10], [11], [12], [13]. These network games, in which players are a network's nodes, are the main focus of this paper.

The classical setting of game theory is static; that is all decisions are made in one step. Conversely, dynamic paradigms of game theory such as evolutionary game theory and repeated games focus more on the dynamics of changes in player actions, i.e., how games evolve. These paradigms have proven to be invaluable in helping to explain many complex and challenging aspects of different disciplines [14], [15], [16]. Best response dynamics is one of the simplest and most straightforward game dynamics in which each round a player is chosen, updating his action to the one which is most beneficial for him. Using such game dynamics, players are sequentially best responding to the current game setting [17], [18], [19]. This process continues until it converges to an equilibrium where no player wants to deviate from the current setting, called an output of the dynamics.

In this paper, for a given game on a networked dynamic system, we want to guide the player's actions so as to reach a state which is most desirable. Since the individual steps of the game and its outcome are formed by the decisions made by the selfish nodes, controlling this best response dynamics is a very complicated task. The only way to impact the game is to change some of its global configurations (e.g., setting prices or determining each player's turn). We assume that these configurations can be changed using predetermined global parameters. Obviously these parameters should not

• *The authors are with the Sharif University of Technology, Tehran 11155-8639, Iran. E-mail: {mohammadamin.fazli, abbas.sys, moslem.habibi}@gmail.com, jhabib@sharif.edu.*

Manuscript received 2 Feb. 2017; revised 19 Feb. 2018; accepted 7 Mar. 2018. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: MohammadAmin Fazli.)
Recommended for acceptance by F. Fagnani.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TNSE.2018.2814590

be impacted by the decisions and actions of individual players. We call these global parameters, *control parameters*.

To measure the desirability of a state, we use an objective function. Therefore our target control problem can be described as an optimization problem, where we search for control parameter values maximizing the objective function. The usual methods for solving such classes of problems involve searching for the control parameters' values and simulating their best response dynamics. The control parameter values that lead to the most desirable equilibrium amongst the reached equilibria, in terms of our objective function, are then chosen. These methods have an excruciatingly long runtime as many different values must be examined and subsequently simulated to reach the optimal convergence. This limits their usefulness in many real-world application scenarios.

In this paper we show how a network's structural properties can be used to noticeably decrease the computation time of the above optimization problem. To this end, we propose a method to derive a mathematical model which computes the control parameters directly from the structural properties of network nodes. Using this model, instead of running a time consuming optimization algorithm, we can compute the network's structural properties and use the mathematical model to find a set of parameter values which can lead to the most desirable outcome in terms of the defined objective function. To prove the effectiveness of this method, we try it on different test networks using various learning algorithms and show that in all cases, the desirability of the outcome is acceptable.

1.1 Setting

Before going any further, we must at first formalize our model's setting. A normal form game is defined as a tuple (P, A, U) where:

- $P = \{p_1, p_2, \ldots, p_n\}$ is the set of n players,
- $A = A_1 \times \cdots \times A_n$, where A_i is a finite set of actions available to p_i. Each vector $a = (a_1, \ldots, a_n) \in A$ is called an action profile,
- $U = (u_1, u_2, \ldots, u_n)$, where $u_i : A \to \mathbb{R}$ is the utility function for p_i.

Obviously, each player can only change his own action, so for simplicity when dealing with player p_i, the action profile a is written as (a_i, a_{-i}) where $a_{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$, consisting of all the components of a except a_i. Thus the utility function of p_i can be defined as $u_i(a_i, a_{-i})$.

The utility function of each player i, $u_i(\cdot)$, is calculated using a gain function $g_i(\cdot)$ and a loss function $l_i(\cdot)$ as follows:

$$\forall_{a \in A} u_i(a_i, a_{-i}) = g_i(a_i, a_{-i}) - l_i(a_i, a_{-i}).$$

The gain and loss functions have natural economic interpretations and can be computed efficiently. The gain function measures the amount of benefit received by a player due to the nature of the game. The loss function measures the costs incurred by each player for choosing his desired action and is parameterized with cost parameters.

Each player tries to choose an action which maximizes his utility function, i.e., a best response to the actions chosen by other players. We define $BR_i(a_{-i})$ to be the set of actions $x \in A_i$ for which $u_i(x, a_{-i})$ is maximized. An action profile a is called a Nash Equilibrium when

$$\forall_{i \in [n]} : a_i \in BR_i(a_{-i}).$$

In best response dynamics, at each step a player p_i is chosen (according to an activation rule such as uniform randomness), and the player's action profile (a_i, a_{-i}) is updated to (a'_i, a_{-i}), where a'_i is one of the actions which grants the most beneficial deviation for p_i in terms of the utility function u_i, i.e., $a'_i \in BR_i(a_{-i})$. This process continues until the game converges to a Nash-equilibrium.

In this paper we are interested in network games in which the actions of players form a network between them. In such games, the players are the network's nodes and each action profile defines a different network (with a different state). So clearly the utility function maps each possible network to a real value. The players' set of actions consist of those which change this network and its state. The best response of player i changes the network to one with the most desirable structure and state for him, i.e., the network which maximizes the value of u_i.

A best response dynamic for a network game $\mathcal{G}$ on a network N is defined as a process $D_{\mathcal{G},N} = \{N^t\}_{t=0,1,\ldots,\infty}$ where $N^0 = N$ and each N^t for $(t > 0)$ is the network evolving from $N^{(t-1)}$ as follows. One player $p_i \in P$ is chosen based on an activation rule and changes the structure and state of the network according to his best response action.

It remains to define the activation rule. Here, we assume that a subset of nodes are chosen to be active during the game and these players take turns in an arbitrary order. Therefore the activation rule of $D_{\mathcal{G},N}$ is defined by two elements T_D and π_D. $T_D \subseteq P$ is the set of active players (nodes). We assume that only nodes in T_D can be activated during the dynamic. π_D is a permutation of T_D's members (an onto and one-to-one function from $\{0, \ldots, |T_D| - 1\}$ to T_D) which defines the order of nodes. We say that a sequential best response dynamic $D_{\mathcal{G},N}$ has converged when for a $T > 0$, N^T is a Nash-equilibrium. From the definition, we have $N^T = N^{T+1}$.

In this paper, we are trying to assert influence on a best response dynamic $D_{\mathcal{G},N}$. In $D_{\mathcal{G},N}$ each node takes selfish actions. So, the only possible strategy for us is to adjust the global control parameters of $D_{\mathcal{G},N}$ whose values cannot be affected by the network nodes. Using such a strategy we try to make $D_{\mathcal{G},N}$ converge to a more desirable Nash-equilibria whose desirability is measured by an objective function $f : \mathcal{N} \to \mathbb{R}$ ($\mathcal{N}$ is the set of all possible networks), therefore we want to maximize $f(N^T)$ where N^T is the converged Nash-equilibrium of $D_{\mathcal{G},N}$. Here we assume the cost parameters (which are embedded in every node's loss function and are show here by λ_i) and activation rule parameters, i.e., parameters for determining the active set (T_D) and for determining the activation order (π_D), are under our control.

1.2 Our Contribution & Related Works

The context of this paper can be categorized as a control theory problem [20], [21]. The usual objective of control theory is to control a dynamical system (discrete or continuous) in order for its output to follow a desired reference. This theory teaches us how to build a controller for a system. This

controller continuously monitors the output of the system and computes its difference with a desired reference (which is called the error). Using the value of the error, the controller computes the related values of the system parameters and adjusts them in order to close the gap between the output and the desired reference. This field has long been of interest to many researchers and has a rich literature [22], [23].

One of the most relevant classes of control theory problems to the overall setting outlined in this paper is called optimized control theory [24], [25], [26]. This theory reduces a control problem to an optimization one with a cost function that is a function of the system state and control variables. An optimal solution to this problem leads to a control policy. Use of optimization algorithms, such as genetic algorithms, is a routine method for computing the parameters of the control policy [27], [28], [29]. Such methods must check a great number of different parameter values. In our problem context, running such checks requires very time-consuming simulations, because to compute the resulting objective function for each set of parameter values, the simulation needs to converge to a Nash-equilibrium. Thus, such existing methods are impractical in even simple simulation scenarios over small networks and are not applicable in many real world situations, specially when we need instant controller responses.

As an alternative one can investigate the intuitive relations between the solution context and the properties of the network to conduct the computations needed for faster obtainment of the optimal solution. Such an intuition has been used in many works in different fields such as complex networks, graph theory and operations research [30], [31], [32], [33]. One of the relevant aspects of dynamical systems on complex networks which has been studied with this approach is their controllability. The controllability of a dynamical system is defined as the number of required control signals to drive the system towards some desirable state. Studies have found correlations between controllability and the structural properties of complex networks [1], [2], [34]. Thus one can use these findings to find optimized control signals in shorter times or enhance the controllability of dynamical systems by some network structural perturbations [3].

In the context of this paper, because multiple agents are selfishly interacting, the complexity is high and finding the relation between optimal values and network properties is difficult. We show how learning algorithms can be used to find such hidden relationships between optimal control parameter values and network properties and derive a mathematical model to map them. This idea can also be used for other network optimization algorithms where no intuitive relation between the solution context and network parameters exists.

Another related field to our work is the field of Mechanism Design in classical game theory and economics [35], [36], [37]. This field deals with games of private information in which each player has a secret information (called its type). A player (that is called the principal) designs the game mechanism based on some objective functions (e.g., social welfare or truthfulness). The players report their type (truthfully or not) and the mechanism is executed, benefiting each player based on his reports and the designed mechanism.

We present our framework in Section 2. As discussed earlier, the aim of this framework is to learn a mathematical model which maps network nodes' structural properties such as degree and clustering coefficient to the nodes' control parameters. We call this mathematical model the NSP2CP model. We test different learning algorithms and show that all of them are effective in finding hidden relationships between the control parameters and network properties. Then we show the efficiency of these models by simulating different best response dynamics (see Section 3) over different networks with the parameters generated by the model.

2 Framework

In this section, we propose a framework to derive the NSP2CP mathematical model which computes control parameters from the structural properties of an input network. The outline of this framework is shown in Fig. 1. The main intuition behind the NSP2CP framework is that if we can derive a mathematical model which relates the structural properties of a given network (called the training network) to the optimized control parameter values, based on inductive reasoning, we can broaden its application to other networks with similar properties. Therefore we can inductively extend this relation to other networks and use it to find their best control parameter values. This is based on previous research showing that many complex networks share similar structural properties [38], [39]. Since in our problem multiple agents are selfishly interacting, we are faced with high complexity and therefore finding the relation between the optimal values and network properties is a complicated task. Hence, instead of directly probing this complicated relation, we use learning algorithms to extract it, calling this the NSP2CP mathematical model.

We first model each control parameter by normally distributed random variables. We assume that each cost parameter λ_i is sampled from a normal distribution with average μ_i^λ and standard deviation σ_i^λ. T_D (the set of active nodes in the best response dynamics) is modeled by a n-bit binary string $\delta = \delta_1\delta_2\dots\delta_n$ for which each

$$\delta_i = \begin{cases} 1 & \text{if } v_i \in T_D \\ 0 & \text{if } v_i \notin T_D. \end{cases}$$

To compute δ_i, we pick a random sample δ' from a normal distribution $\mathcal{N}^\delta$ with average μ_i^δ and standard deviation σ_i^δ and put $\delta_i = 1$ if and only if $\delta' > 0.5$. Finally, $\pi_D = \pi_1\pi_2\dots\pi_n$ (the order of players' activation) is a permutation which maps each network node to a unique order from $\{1, 2, \dots n\}$. We model this permutation with a matrix θ whose entries $\theta_{i,j}$ are defined as

$$\theta_{i,j} = \begin{cases} 1 & \text{if } \pi_D^{-1}(v_i) < \pi_D^{-1}(v_j) \\ 0 & \text{if } \pi_D^{-1}(v_i) > \pi_D^{-1}(v_j). \end{cases}$$

We compute $\theta_{i,j}$ with the same method used for δ_i, by sampling from a normal distribution $\mathcal{N}^\theta$ with average $\mu_{i,j}^\theta$ and standard deviation $\sigma_{i,j}^\theta$.

Our goal is to obtain a mathematical model (like a closed formula or the neural network shown in Fig. 1), mapping network structural properties to control parameters. For example the formula shown in this figure calculates $\sigma_{i,j}^\theta$ as a function of the betweenness of nodes i and j, based on the

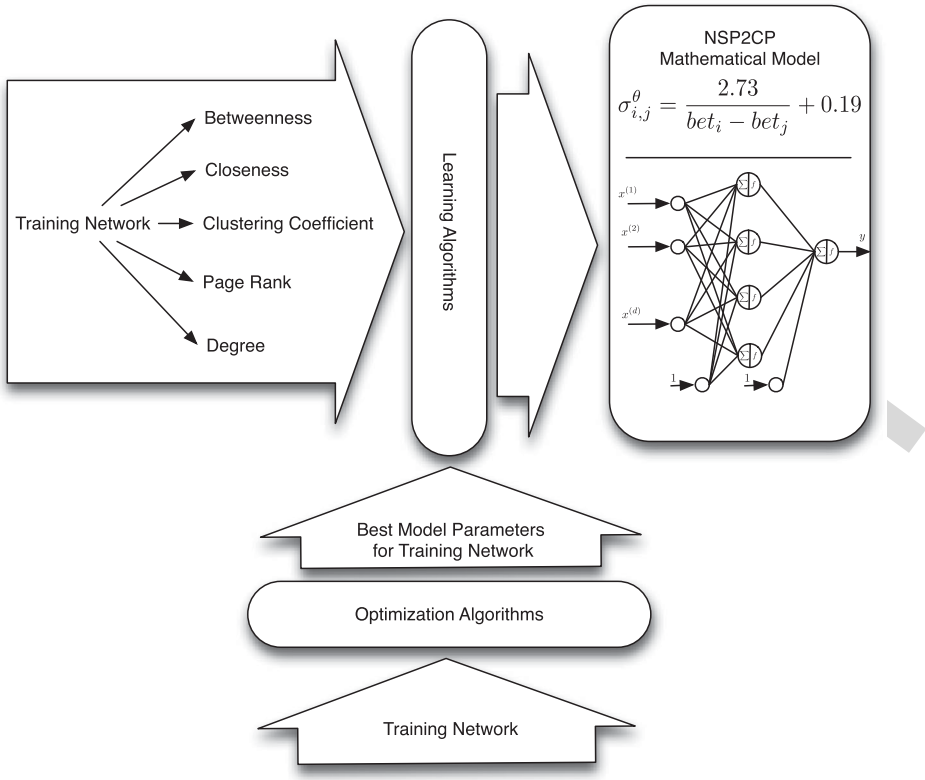

Fig. 1. Learning the NSP2CP mathematical model from the training network's structural properties.

results of our framework which will be discussed below. The framework starts with a training network N_{tr}. In the first step, the structural properties of N_{tr} are computed. In this paper, we consider five different properties: betweenness, closeness, clustering coefficient, page rank and degree. Thus, for each node v_i of N_{tr}, we have a 5-component vector $P_i = < p_{i,1}, p_{i,2}, \ldots, p_{i,5} >$, where each $p_{i,j}$ determines one of the computed structural properties of node v_i in N_{tr}.

In the next step, an optimization algorithm is run to search for the optimized control parameters of the best response dynamics over N_{tr}. Assuming that N_{tr} has n nodes, $\binom{n}{2} + 2n$ control parameters must be computed in this step, which includes λ_i and δ_i for $1 \leq i \leq n$, and $\theta_{i,j}$ for $1 \leq i < j \leq n$. Thus, to solve this problem, we must search in a high-dimensional space where checking each point requires a lenghty simulation. Hence, with a large size training network this optimization is infeasible. Therefore N_{tr} should be a small network with diverse node properties. To search for the optimal control parameter values, we use a hybrid algorithm combining genetic and hill-climbing algorithms [40] which will be introduced in Section 2.1.

Next, we use supervised learning algorithms to extract the NSP2CP mathematical model. To do this, for each of the parameters which correspond to a single node i.e., μ^{λ}, σ^{λ}, μ^{δ} and σ^{δ}, we construct a set of training data with n members, each of which is a pair consisting of one of the P_is as input and the optimal value for the target parameter for the ith node as the supervisory signal. For example, the training dataset for μ^{λ} is formed as follows:

$$\{ < P_1, opt(\mu^{\lambda}_1) >, < P_2, opt(\mu^{\lambda}_2) >, \ldots, < P_n, opt(\mu^{\lambda}_n) > \},$$

where $opt(\mu^{\lambda}_i)$ is the optimal value for μ^{λ}_i in the training network computed by the optimization algorithm. To learn, $\mu^{\theta}_{i,j}$ and $\sigma^{\theta}_{i,j}$ which are related to two nodes, we use both P_i and P_j as the input object, for example the training dataset for $\mu^{\theta}_{i,j}$ has the form

$$\{ < P_i, P_j, opt(\mu^{\theta}_{i,j}) > \}_{1 \leq i < j \leq n}.$$

The training dataset is fed to the learning algorithms to derive the NSP2CP mathematical model. In this paper, we test two different learning methods for this purpose: linear regression (LR) and multilayer perceptron (MLP) [41], described briefly in Section 2.2. The output of these methods is a mathematical formula or a neural network which map nodes' structural properties to control parameter values. From now on, we can use this model for other (test) networks (Fig. 2). We only need to compute the structural properties of each node and feed them to the models to compute the control parameters. In Section 3, we show the efficiency of these parameters.

2.1 Optimization Algorithm

For the optimization algorithm, we use a hybrid method combining hill-climbing and genetic algorithms. These meta-heuristic methods were invented to limit the search space and reduce the runtime in exchange for losing the assurance of obtaining the exact optimized value. Although they have a much smaller runtime in comparison to brute-force mechanisms, computing the objective function value for each combination of parameter values in the search space needs a complete simulation of the best response dynamics until it converges to an equilibrium, which can

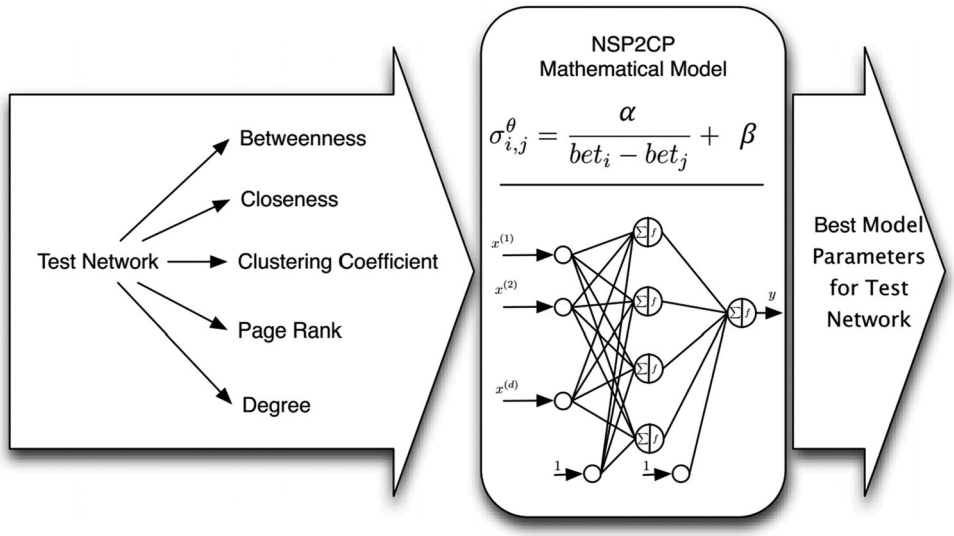

Fig. 2. Computing a test network's control parameters from the NSP2CP mathematical model.

take a long time, even on our chosen small scale training network. For the Hill-climbing algorithm, we start from a feasible solution, a point in our search space, and then iteratively move to one of its neighbors which locally maximizes the objective function. We continue this process until we reach a point called a local maximum, where none of the available neighbors can grant a higher value to the objective function then the one being provided by the current point. One of these local maximum points is the global maximum. It is a good technique to run the whole process with random starting points (random restarts) to increase the probability of reaching the global maximum.

We use a hill-climbing algorithm with 10 random restart to find optimized values for λ_i variables. Thus our target point is a n-dimensional vector with the form $\lambda = <\lambda_1, \lambda_2, \ldots, \lambda_n>$. We run this hill-climbing algorithm 30 times, each time setting T_D equals to the set of all nodes and π_D to a random permutation. We consider $2n$ neighbors for each point λ, where neighbors are obtained by increasing or decreasing the values of components of λ by $\alpha = 0.5$. After running all instances of the hill-climbing algorithm, we will have 30 n-dimensional vectors for λ. By calculating the mean and standard deviation of the components of these vectors, we will have the values for μ_i^{λ} and σ_i^{λ} for each $1 \leq i \leq n$.

In genetic algorithms, a population of candidate solutions (points) is evolved toward better solutions. This evolution is an iterative process where in each iteration, 3 operators are applied: mutation, crossover and selection. Applying mutation and crossover operators generates new solutions and adds them to the current population. The mutation operator creates a copy from one of the solutions and does some minor modifications on it. The crossover operator picks two solutions and generates a new child solution by combining their properties. The selection operator reduces the population by selecting some of the current population for the next iteration and ignoring the others. This is done in a way that solutions with a higher objective function value are more likely to be selected and transferred to the next iteration.

We use a genetic algorithm to search for optimized values for π_D parameters. For π_D, we generate 100 random permutation as the initial population, each encompassing a potential activation order for the nodes. In each iteration, we pick 50 of the current population randomly and mutate them to generate 50 new solutions. We also select 50 random pairs from the population and generate 50 new solutions by their crossover. After these steps, we will have a population of 200 (100 from the previous iteration, 50 mutated solutions and 50 child solutions). Finally, we select 100 of the solutions for the next iteration so that solutions with a higher objective function value are more likely to be selected. After the convergence of the population, where the maximum objective function value of the population doesn't change in two consecutive iterations, we transform all the populations to their matrix form ($[\theta_{i,j}]$) and calculate their mean and standard deviation to achieve $\mu_{i,j}^{\theta}$ and $\sigma_{i,j}^{\theta}$. Consider that in all these steps, T_D is equal to the set of all nodes and λ_i is set to the optimized value of μ_i^{λ} for each $1 \leq i \leq n$ computed in the previous phase.

For the mutation operator, we pick two random nodes and change their order in the input permutation. That is, if the input permutation is $\pi = \pi_1, \ldots, \pi_i, \ldots, \pi_j, \ldots, \pi_n$ and the random nodes picked are π_i and π_j, the mutated permutation will be $\pi' = \pi_1, \ldots, \pi_j, \ldots, \pi_i, \ldots, \pi_n$. The crossover operator gets two permutations π^1 and π^2 as input. To generate the child permutation π', it selects a random subset R of nodes with $\frac{n}{2}$ members. The nodes in R will appear in the first $|R|$ positions of π' with the order defined by π^1. The remaining $n - |R|$ nodes will appear in the last positions of π' by the order defined by π^2.

To find the optimized values of T_D's related parameters, we use a genetic algorithm with the same approach we used for π_D. The search space consists of all n-bit binary strings. The only difference is the mutation and crossover operators. The mutation operator gets a binary string as input, chooses a random bit and changes it to its negation. The crossover operator gets two bit strings T^1 and T^2 and selects a random subset R of $\{1, 2, \ldots, n\}$ with $\frac{n}{2}$ members. Then it computes the output bit string T by choosing the values of locations in R from T_1 and others from T_2.

2.2 Learning Methods

As was mentioned earlier, in this paper we use two learning methods, RL and MLP. In this section, we briefly explain these learning methods. Learning algorithms are tools to build an abstract model from an example input (which is called the training data) in order to make predictions and decisions. These models give us a general view of data, which cannot be captured by looking at them separately. The main approach of these algorithms is to first select a mathematical model which intuitively has the ability to explain the system behavior and then fit the observed data into it. The fitting is done by defining an error function which shows the deviation between the model and the data and then by calibrating the model parameters to minimize the error function.

Throughout this section we assume that we have n training data, each represented by a $d+1$-dimensional vector. Each of these training data is in the form $<x_i, y_i>$, where x_i is a d-dimensional vector called the label and y_i is its corresponding target value. The desired learned model $\mathcal{M}$ should map each x_i to y_i. For example in our problem, x_is are the vectors containing network structural property values and y_is are control parameters.

The linear regression method fits the training data into a linear model

$$\mathcal{M}(x) = w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \cdots + w_d x^{(d)},$$

where $x^{(j)}$ is the jth component of x. The LR algorithm mainly tries to find w_is that minimize the following error function by optimization techniques such as gradient descent

$$\sum_{i=1}^{n} (\mathcal{M}(x_i) - y_i)^2 + \frac{\zeta}{2} \sum_{i=0}^{d} w_i^2.$$

The first part of the above formula is the squared error function. The second part is called the regularization part and ζ is called the regularization parameter which is used to avoid over fitting the problem [41]. This model can be extended by changing variables through applying different kinds of functions, called basis functions, on the training data. For example applying $\phi(z) = \frac{1}{z}$ on y_is and keeping x_is without variable change, will lead to the following model:

$$\mathcal{M}(x) = \frac{1}{w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \cdots + w_d x^{(d)}}.$$

A multilayer perceptron is a feedforward artificial neural network which consists of a set of interconnected computational blocks which are called neurons. Each neuron does a simple computation. Assuming that one of the neurons $n^{(i)}$ has $k+1$ inputs $I_0 = 1, I_1, I_2, \ldots, I_k$, it computes a weighted linear combination of these inputs and then applies an activation function (which is usually a logistic sigmoid function $\sigma(z) = \frac{1}{1+exp(-z)}$) on them, activating the output when the input has the enough amount of amplitude. Thus O^i, which defines the output of $n^{(i)}$, is computed as follows:

$$O^{(i)} = \sigma\left(\sum_{j=1}^{k} w_j^{(i)} I_j\right),$$

where $w_j^{(i)}$ is the weights defined for $n^{(i)}$. Although the modeling power of each neuron is very low, by networking them we will obtain a very powerful model, specially when a hidden layer is considered in its structure. To learn an MLP model, we should find optimum neuron weights $w_j^{(i)}$ in order to minimize the squared error function. The back-propagation method is an efficient technique devised for this purpose [41].

3 Evaluation

In this section, our goal is to evaluate the performance of the NSP2CP framework. For this purpose, first we choose some network games and formally define in our settings. Then the evaluation methodology is described and finally the results are presented and analyzed.

3.1 Target Network Games

In this section, we describe three common network games and define them formally in our settings. These network games are used as a basis for our experiments in further sections.

3.1.1 Community Formation Games

Understanding the formation and evolution of overlapping communities in networks is one of the active lines of research in network sciences which has applications in sociology, criminology, social marketing and many other fields [42], [43], [44]. In order to study issues such as these in multi-agent and dynamical settings, community formation games have been proposed [10], [42].

In this paper, we consider the community formation game proposed by Chen et al. [10] which uses Newman's modularity function [45] as the nodes' gain function. Here, nodes like to increase the number of their inter-community relations and decrease the number of their intra-community relations. We assume that, on the other hand, we don't want the network to converge to a collection of separated communities, preferring the network structure that allows high speed dissemination of information between communities. We now formalize all these aspects in our setting.

The best response dynamic for this community formation game on a network N is stated as follows. Each node v_i chooses a set of communities $c_i \subseteq C$ and enrolls in them. His aim is to maximize his local modularity, which shows the degree to which v_i has inter-community interactions in comparison to intra-community interactions. In the work done by Chen et al. [10] the following gain function for node v_i, when his action is c_i, is proposed

$$g_i(N) = \frac{1}{m} \sum_{v_j \in V_N} \Big(A(v_i, v_j)\delta(v_i, v_j) - \frac{deg_N(v_i) deg_N(v_j)}{2m} |c_i \cap c_j| \Big),$$

where

- m is the number of N's edges,
- $A(v_i, v_j) = \begin{cases} 1 & \text{there is an edge between } v_i \& v_j \\ 0 & \text{o.w.} \end{cases}$
- $\delta(v_i, v_j) = \begin{cases} 1 & \text{if } c_i \cap c_j \neq \emptyset \\ 0 & \text{o.w.} \end{cases}$

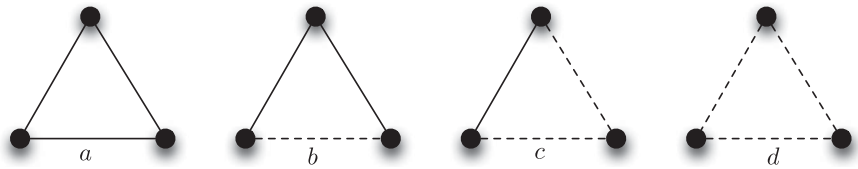

Fig. 3. Possible undirected triad in a signed network. Full and dashed lines represent positive and negative relations respectively.

The loss function of each node v_i is simply defined as $l_i(N) = \lambda_i |c_i|$ where λ_i is the cost of enrolling v_i in each community.

As mentioned before, we assume that we want to increase the speed of information dissemination between communities, which can be measured by computing the closeness of these different communities to each other. To do so, we first build a weighted complete graph $G_C = (V_C, E_C)$ from the Nash-equilibrium network N, created from an action profile using the following procedure:

(1) Define $V_C = \bigcup_{v_i \in V_N} c_i$
(2) For each $p, q \in V_C$, the weight of the edge between p and q in G_C is set to $\frac{1}{e_{p,q}+2v_{p,q}}$ where $e_{p,q}$ is the number of edges in N that exist between nodes enrolled in p and q and $v_{p,q}$ is the number of N's nodes which are enrolled in both p and q.

$f(N)$ is computed from G_C by the following formulation:

$$f(N) = \frac{1}{|V_C|}\left[\sum_{p,q \in V_C} dist_{G_C}(p,q)\right]^{-1},$$

where $dist_{G_C}(p,q)$ is the weighted shortest path between p and q in G_C.

3.1.2 Network Creation Games

Complex networks have known structural properties such as low average distance, high clustering coefficient and power-law degree distribution. Many researchers have concentrated on building network models which capture these properties [46]. Network creation (formation) games have been devised as a multi-agent model for this aim [47], [48], [49]. In this paper, we consider the network creation game proposed by Brautbar and Kearns [11] in which nodes try to maximize their clustering coefficient. The clustering coefficient of a node is the proportion of edges that exist between the node and neighboring vertices to the number of links that could possibly exist between them.

Each node v_i can decide on the existence of his neighboring edges. The gain function of node v_i is defined as

$$g_i(N) = \frac{\Delta_N(v_i)}{\binom{deg_N(v_i)}{2}}.$$

The loss function is simply the cost of purchasing edges, i.e.,

$$l_i(N) = \lambda_i \cdot deg_N(v_i),$$

where λ_i is the cost of purchasing each edge for v_i.

For this game, our objective function is defined so as to capture the fact that we want the nodes of the Nash-equilibrium network as close as possible. Thus for a Nash-equilibrium N of this game we define $f(N)$ as N's closeness

$$f(N) = \frac{1}{n}\sum_{v_i, v_j \in V_N} \frac{1}{dist_N(v_i, v_j)},$$

where $dist_N(v_i, v_j)$ is the length of the shortest path between v_i and v_j in the network N.

3.1.3 Signed Network Formation Games

Signed networks are networks in which edges are labeled as either negative or positive. The semantic of these labels is related to the context in which the network is defined. Signed network formation games have been defined similar to network creation games as a way to study the evolution of signed networks. The signed network formation game that we focus on here is based on the theory of structural balance in networks [50]. Structural balance theory describes attitudes of individuals to reduce cognitive dissonance among each other. When nodes set up dyadic relations that contain both positive and negative interactions, four different types of triad relations can be created (Fig. 3). In conformity to this theory, we can classify these triangles into 2 classes: balanced and unbalanced. Triads (a) and (c) are balanced and relatively stable, but triads (b) and (d) are unbalanced and susceptible to break apart.

Each node v_i can decide on the existence and subsequently sign of his neighboring edges. The node's utility is defined as a linear combination of the number of balanced triangles and unbalanced triangles he is involved in

$$g_i(N) = \Delta_N^+(v_i),$$

and

$$l_i(N) = \lambda_i \cdot \Delta_N^-(v_i),$$

where

- $\Delta^+(v)$ is the number of balanced triangles of N v_i is involved in.
- $\Delta^-(v)$ is the number of unbalanced triangles of N v_i is involved in.
- λ_i is a parameter which defines a cost for nodes when they contribute to an unbalanced triad.

Nodes tend to have more balanced relations, so they try to manage their relationships in order to maximize the number of balanced triads they are involved in, but stable signed networks can have some unpleasant properties. It has been shown that when networks have no unbalanced triad, its nodes can be partitioned into two subsets wherein edges in the same subset are all positively labeled and the edges between different subsets are negatively labeled [51]. This creates a bipolar structural with two rival groups of united nodes which are enemies of each other, which in many

TABLE 1
Properties of the Training Network

Property	Average	Standard Deviation
Degree	4.6	3.940
Clustering Coefficient	0.11	0.578
Betweenness	46.47	99.270
Closeness	0.129	2.18×10^{-3}
PageRank	0.029	0.022

TABLE 2
Test Networks

Test Network	#Nodes	#Edges	Reference
Dolphin	62	318	Lusseau et al. [53]
Jazz	198	5,484	Heckathorn et al. [54]
Netscience[1]	379	1,827	Newman [55]

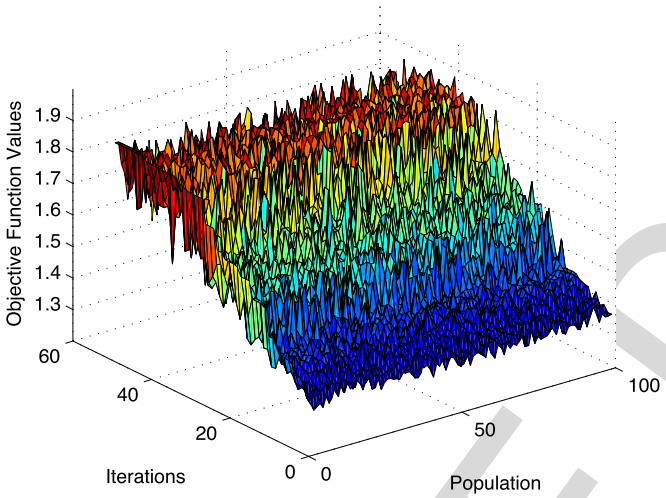

Fig. 4. Our genetic algorithm implemented for optimizing the μ_i^δ control parameters for the community formation game.

contexts is not a desired property. One of the best analysis of World War I is based on the formation of a such bipolar structure in international relationships between the six main warring countries [5]. Thus here we want the Nash-equilibrium network N as unstable as possible, so we define $f(N) = \frac{\Delta_N^-}{\Delta_N^- + \Delta_N^+}$, where Δ_N^- and Δ_N^+ are the number of unbalanced and balanced triangles in N respectively.

3.2 Training & Test Networks

In our framework, nodes' structural properties (degree, clustering coefficient, betweenness, closeness and pagerank are chosen) are used for learning control parameters. For this purpose, we first should choose a training network and we should focus on these properties. As we previously saw in Section 2, our training network must be small and contain diverse node properties. With this in mind, we chose Zachary's karate club-network [52] with 34 nodes and 78 edges as the training network. The properties of this network are depicted in Table 1.

For testing our framework we should choose different test networks. Since we want to run the optimization algorithm on these networks to compare our framework with it (the best possible output) and this algorithm is very time consuming, test networks should be small. It is important to notice that running this algorithm even on these small networks takes many hours. So three small networks which are important in social network research are chosen as our test networks (See Table 2).

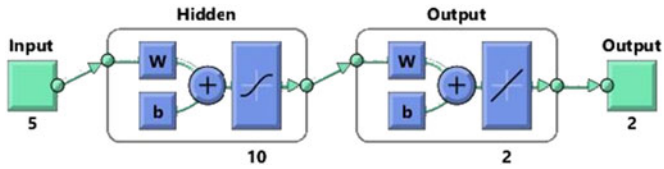

Fig. 5. The MLP model trained for the μ_i^δ and σ_i^δ variables.

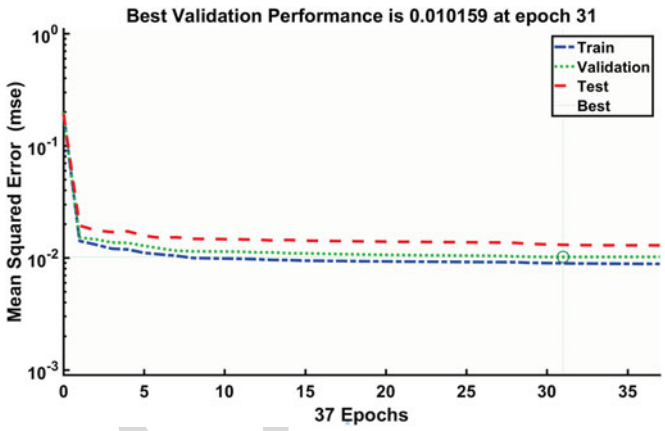

Fig. 6. The convergence diagram for the MLP shown in Fig. 5.

3.3 Simulations & Results

In Section 3.1, we choose three different network games to evaluate the NSP2CP framework. So for each of these games, the framework's mathematical models must be learned. To do this, as it is depicted in Fig. 1, we should first run the optimization algorithm (see Section 2.1) on the training network. The output of this long-time running algorithm is the best set of control variable values we can achieve for the training network's nodes. In Fig. 4, we have shown an example for the behaviour of the genetic algorithm used for finding the best values for μ_i^δ control parameters. This figure shows how the objective function values for the population in the community formation game are effectively increasing during this phase until they converge to their optimums.

In the next step of our framework (See Fig. 1), the structural properties of the training network's nodes are computed and alongside with the output of the optimization algorithm, are fed into a learning algorithm (see Section 2.2). This learning algorithm should calculate a model for the fast computation of control variables. For implementing these learning algorithms Matlab's standard toolboxes are used. For example for learning the MLP model, the Neural Network toolbox is used which gets the number of neurons in the hidden layer (we used 10) and then builds and trains the desired MLP model. The resulting trained network for calculating activation parameter variables is depicted in Fig. 5. This network is trained for 31 epochs where the validation error reaches its least value. The convergence diagram during this phase is depicted in Fig. 6.

Finally, for each network game and each test network, we consider the following test scenarios and compare their

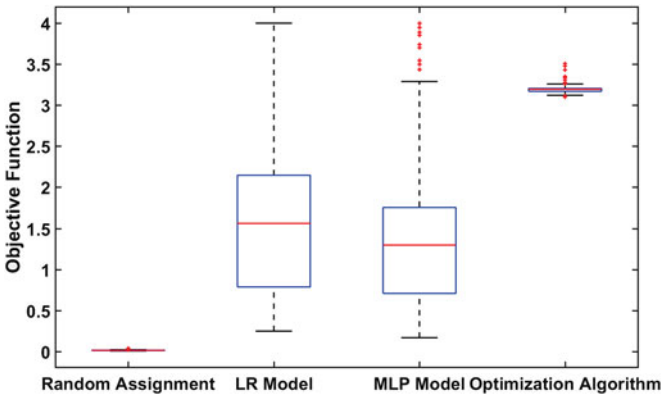

Fig. 7. Testing NSP2CP for the community formation game on Dolphin network.

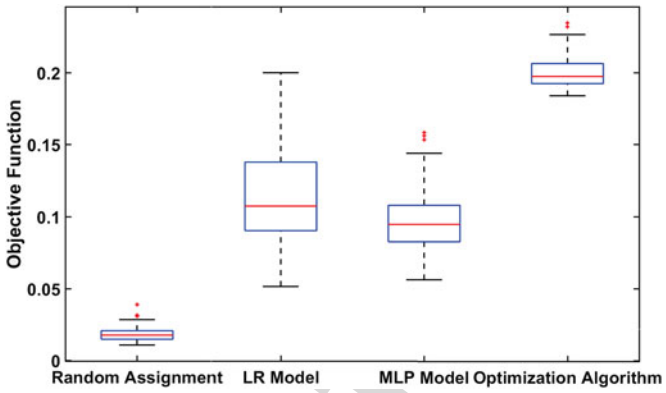

Fig. 8. Testing NSP2CP for the community formation game on Jazz network.

results. Each scenario is run using a different method. Among these methods, the LR and MLP models are derived from the NSP2CP framework. As described in Section 2, in these scenarios the trained mathematical models are used to compute the control signals. These mathematical models (a closed formula or a neural network) can compute the control signals from the structural properties of each test network instantly.

- LR Model: the value of control parameters are computed from the LR model. The LR model is a set of closed formulas which map the structural properties of nodes to the average and standard deviation of control parameters. These formulas are obtained by the linear regression algorithm.
- MLP Model: the value of control parameters are computed from the MLP model. This model is a set of learned neural networks which take the value of nodes' structural properties and compute average and standard deviation of control parameters. These neural networks are learned by the multilayer perceptron algorithm.
- Optimization Algorithm: the value of control parameters are computed by running our optimization algorithm on the test network. Running an optimization algorithm for each instance of the problem (for each test network) is a typical but time consuming way to compute the control parameters. It is trivial that the control parameters computed by this method are always better than the previous two methods, because the LR and MLP models are learned from the output of this model. Since for the best response dynamics there are many such parameters, using this method for controlling them is very slow and therefore infeasible. Our goal for running this scenario is to compare the optimality of this method with NSP2CP driven algorithms.
- Random Assignment: the value of control parameters are chosen randomly. One can hypothesize that the optimality and effectiveness of the LR and MLP models can be by chance, i.e., any random assignment to the control parameters can result in such optimality. This scenario is used for comparison and to reject this hypothesis.

We run 100 simulations (on a machine with 8 GB of RAM and a Core i3 processor running at 3.3 GHz) for each of the above scenarios (methods) for each network game on each test network. For each simulation, we derive the value of the control signals by sampling from the normal distributions whose mean and standard deviations are determined based on the chosen scenario.

While the optimization algorithm scenario needs a very long time to converge and find optimal control parameters, in the LR and MLP scenarios the control parameters are calculated instantly. The charts in the paper show that although these models are very fast, their outcomes are comparable to the time intensive optimization algorithms. We use box plots to compare the values of the objective function in different scenarios. The results are shown in Figs. 7, 8, 9, 10, 11, 12, 13, 14, and 15. Each figure is related to one of the games introduced in Section 3.1 and one of the test networks (See Table 2). As can be seen from the box plots, the results of our proposed model compare favorably to the results that can be achieved using the optimization algorithm, which takes many hours to complete. So whilst our results behave much more optimally than the random assignment scenario and also in many cases are close to the optimized algorithm, we have nearly instantaneous output of the control parameter values.

By the above simulation scenarios, we show that using a learning algorithm to calculate control signals from the network structural properties has a tremendous effect on the time and optimality of the control signals. We still have to face a new question: which learning algorithm? Our experiment shows that it only depends on the game. As it can be

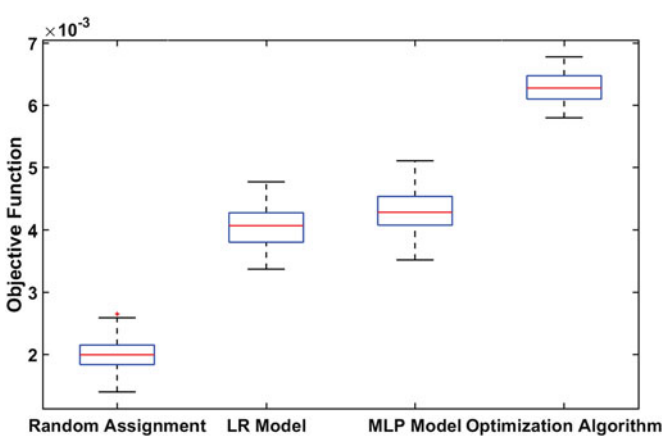

Fig. 9. Testing NSP2CP for the community formation game on Netscience network.

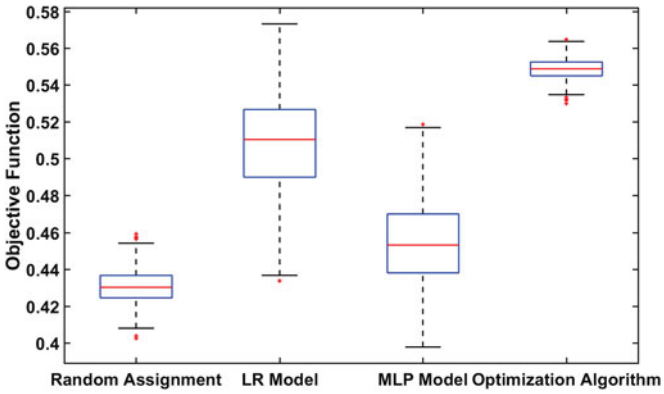

Fig. 10. Testing NSP2CP for the network creation game on Dolphin network.

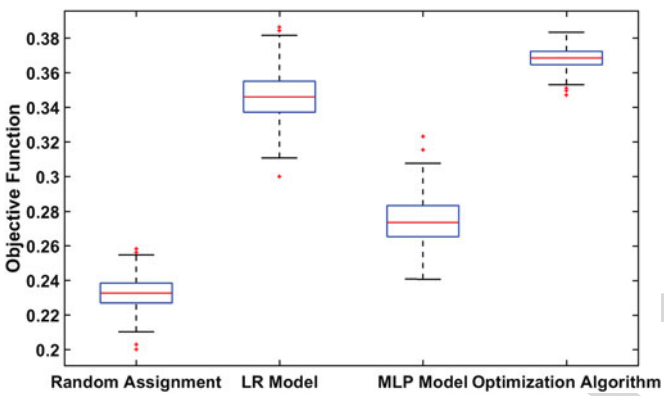

Fig. 11. Testing NSP2CP for the network creation game on Jazz network.

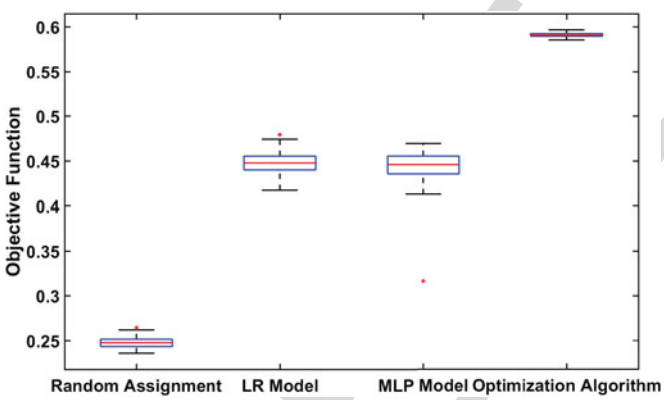

Fig. 12. Testing NSP2CP for the network creation game on Netscience network.

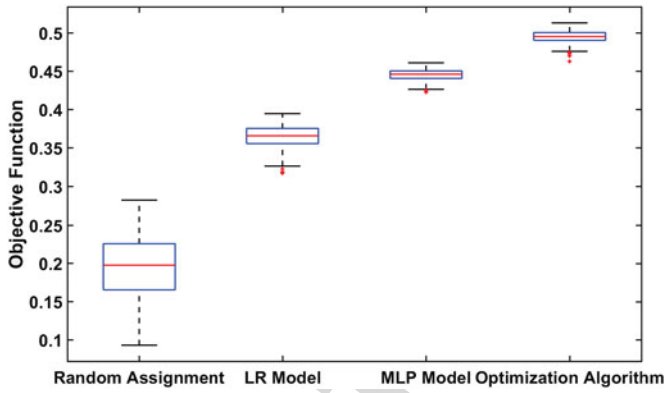

Fig. 13. Testing NSP2CP for the signed network formation game on Dolphin network.

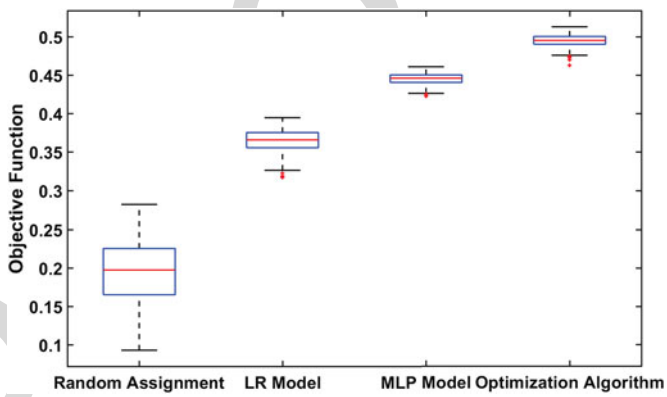

Fig. 14. Testing NSP2CP for the signed network formation game on Jazz network.

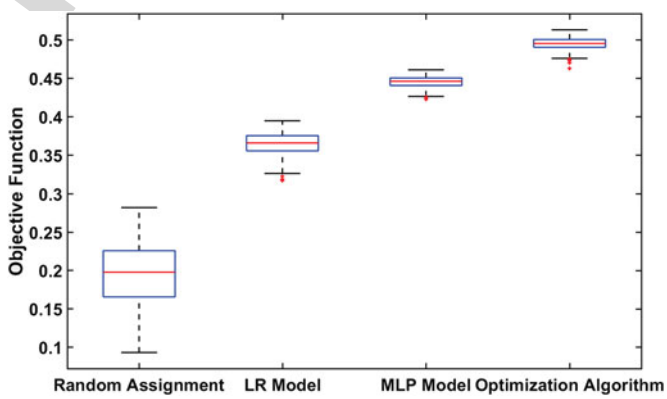

Fig. 15. Testing NSP2CP for the signed network formation game on Netscience network.

seen from the Figs. 7, 8, 9, 10, 11, 12, 13, 14, and 15 for all three test networks, the linear regression algorithm has better performance for the community formation and the network creation games. On the other side, the multilayer perceptron model is better for the signed network formation game for all three test networks. This may be because of the opposite directions of the utility and the objective function in this game, i.e., if a node increases her utility function, she may decrease the objective function.

4 Conclusion

Controlling complex dynamical systems involving many agents is a complex and challenging problem arising in many fields, specially when the involved agents are rational and selfish. A classical but time consuming method for calculating control signals in such systems is to use optimization algorithms to find best values for these parameters. The problem is that in many scenarios when the number of control signals increase the running time will also increase exponentially, specially when meta-heuristics are used.

In this paper, we focus on one of the simplest game theoretical dynamical systems called best response dynamics whose agents are connected through a network. Since such systems are very complex, controlling them involves a high dimensional search space needing an optimization algorithm with a large run time. Another problem when dealing

with such systems is that even with a given set of parameter values, calculating the objective function requires a lengthy simulation, where probing each point is very time-consuming. Taking these two problems into account, classical methods are impractical and inefficient in many instances.

In this paper the NSP2CP framework was introduced which uses the relation between structural properties of a given network and control parameters to derive values for the control parameters. As was shown, our method yields a good enough result in a fraction of the time of classic methods. In this framework, we use typical learning algorithms whose main goal is to find patterns in data and abstract it by mathematical formulations.

Our approach can be used in other researches which try to study different properties of complex networks, specially when it is desirable to find correlations between them. With learning algorithms, one can not only find complex correlations but also extract an exact formulation which can also be used for other important goals such as optimizations and analytical studies.

As far as we know, this paper is the first which considers controlling game theoretical dynamics over networks. Many follow-up works can be proposed for this paper which are interesting and also have applications in different fields:

- We have focused on best response dynamics which simply assumes that players have no memory and do not become experienced when the game goes on. Considering more natural dynamics [56], [57] which considers players with learning powers are more challenging and also more interesting.
- It has been shown that there exists a subcategory of games in which computing the best response is NP-Hard [12], [58], thus even a computer with high computing power can not feasibly find the solution. Players in new game theoretical models are satisfied with approximate best responses which are not optimal but are guaranteed to be near the optimal value [59], [60]. Controlling the dynamics defined over these models can be an interesting follow-up.
- In our model we used a limited setting including a small training network and a small number of structural properties. We can predict that better results can be achieved by using this framework with extended settings.

REFERENCES

[1] M. Egerstedt, S. Martini, M. Cao, K. Camlibel, and A. Bicchi, "Interacting with networks: How does structure relate to controllability in single-leader, consensus networks?" *IEEE Control Syst.*, vol. 32, no. 4, pp. 66–73, Aug. 2012.

[2] M. Pósfai, Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Effect of correlations on network controllability," *Sci. Rep.*, vol. 3, 2013, Art. no. 1067.

[3] W.-X. Wang, X. Ni, Y.-C. Lai, and C. Grebogi, "Optimizing controllability of complex networks by minimum structural perturbations," *Phys. Rev. E*, vol. 85, no. 2, 2012, Art. no. 026115.

[4] J. Sun and A. E. Motter, "Controllability transition and nonlocality in network control," *Phys. Rev. Lett.*, vol. 110, no. 20, 2013, Art. no. 208701.

[5] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[6] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[7] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[8] M. Osborne, *An Introduction to Game Theory*," Oxford Univ. Press, London, U.K., 2004.

[9] D. Foster and P. Young, "Stochastic evolutionary game dynamics," *Theoretical Population Biol.*, vol. 38, no. 2, pp. 219–232, 1990.

[10] W. Chen, Z. Liu, X. Sun, and Y. Wang, "Community detection in social networks through community formation games," in *Proc. Int. Joint Conf. Artif. Intell.*, 2011, Art. no. 2576.

[11] M. Brautbar and M. Kearns, "A clustering coefficient network formation game," in *Algorithmic Game Theory*. Berlin, Germany: Springer, 2011, pp. 224–235.

[12] M. Malekzadeh, M. Fazli, P. J. Khalilabadi, H. Rabiee, and M. Safari, "Social balance and signed network formation games," in *Proc. KDD Workshop Social Netw. Anal.*, 2011.

[13] I. Menache and A. Ozdaglar, "Network games: Theory, models, and dynamics," *Synthesis Lectures Commun. Netw.*, vol. 4, no. 1, pp. 1–159, 2011.

[14] C. Cleveland, D. Liao, and R. Austin, "Physics of cancer propagation: A game theory perspective," *AIP Advances*, vol. 2, no. 1, 2012, Art. no. 011202.

[15] M. Asano, I. Basieva, A. Khrennikov, M. Ohya, and Y. Tanaka, "Quantum-like dynamics of decision-making in prisoner's dilemma game," in *Proc. Found. Probability Phys.*, 2012, pp. 453–457.

[16] M. Kandori and S. Obayashi, "Labor union members play an OLG repeated game," *Proc. Nat. Academy Sci. United States America*, vol. 111, no. Supplement 3, pp. 10 802–10 809, 2014.

[17] L. DallAsta, P. Pin, and A. Ramezanpour, "Statistical mechanics of maximal independent sets," *Phys. Rev. E*, vol. 80, no. 6, 2009, Art. no. 061136.

[18] D. López-Pintado, "Contagion and coordination in random networks," *Int. J. Game Theory*, vol. 34, no. 3, pp. 371–381, 2006.

[19] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden, "The price of stability for network design with fair cost allocation," *SIAM J. Comput.*, vol. 38, no. 4, pp. 1602–1623, 2008.

[20] R. C. Dorf and R. H. Bishop, "Modern control systems," 1998.

[21] Z. Bubnicki, *Modern Control Theory*. Berlin, Germany: Springer, 2005.

[22] E. Fernández-Cara and E. Zuazua, "Control theory: History, mathematical achievements and perspectives," *Bol. Soc. Esp. Mat. Apl.*, 2009.

[23] C. Bissell, "A history of automatic control," in *Springer Handbook of Automation*. Berlin, Germany: Springer, 2009, pp. 53–69.

[24] A. Seierstad and K. Sydsaeter, *Optimal Control Theory with Economic Applications*. Amsterdam, The Netherlands: North-Holland, 1987.

[25] R. Gamkrelidze, *Principles of Optimal Control Theory*. Berlin, Germany: Springer, 2014.

[26] D. E. Kirk, *Optimal Control Theory: An Introduction*. North Chelmsford, MA, USA: Courier Corporation, 2012.

[27] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA, USA: Athena Scientific, 1995.

[28] Z. Michalewicz, J. B. Krawczyk, M. Kazemi, and C. Z. Janikow, "Genetic algorithms and optimal control problems," in *Proc. 29th IEEE Conf. Decision Control*, 1990, pp. 1664–1666.

[29] J. M. Van Ast , R. Babuška, and B. De Schutter, "Ant colony learning algorithm for optimal control," in *Interactive Collaborative Information Systems*. Berlin, Germany: Springer, 2010, pp. 155–182.

[30] Z. Jiang, M. Liang, and D. Guo, "Enhancing network performance by edge addition," *Int. J. Modern Phys. C*, vol. 22, no. 11, pp. 1211–1226, 2011.

[31] V. Latora and M. Marchiori, "Efficient behavior of small-world networks," *Phys. Rev. Lett.*, vol. 87, no. 19, 2001, Art. no. 198701.

[32] A. Beygelzimer, G. Grinstein, R. Linsker, and I. Rish, "Improving network robustness by edge modification," *Physica A: Statistical Mech. Appl.*, vol. 357, no. 3, pp. 593–612, 2005.

[33] G. Chen and Z. Duan, "Network synchronizability analysis: A graph-theoretic approach," *Chaos: An Interdisciplinary J. Nonlinear Sci.*, vol. 18, no. 3, 2008, Art. no. 037102.

[34] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *Nature*, vol. 473, no. 7346, pp. 167–173, 2011.

[35] N. Nisan and A. Ronen, "Algorithmic mechanism design," in *Proc. 31st Annu. ACM Symp. Theory Comput.*, 1999, pp. 129–140.

[36] M. O. Jackson, "Mechanism theory," *Available at SSRN 2542983*, 2014.
[37] J. D. Hartline and B. Lucier, "Bayesian algorithmic mechanism design," in *Proc. 42nd ACM Symp. Theory Comput.*, 2010, pp. 301–310.
[38] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Modern Phys.*, vol. 74, no. 1, 2002, Art. no. 47.
[39] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks: Structure and dynamics," *Phys. Rep.*, vol. 424, no. 4, pp. 175–308, 2006.
[40] S. J. Russell and P. Norvig, "Artificial intelligence: A modern approach (international edition)," 2002.
[41] C. M. Bishop, et al., *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[42] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Comput. Surveys*, vol. 45, no. 4, 2013, Art. no. 43.
[43] D. McKenzie-Mohr, *Fostering Sustainable Behavior: An Introduction to Community-Based Social Marketing*. Gabriola Island, BC Canada: New Society Publishers, 2013.
[44] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E*, vol. 80, no. 1, 2009, Art. no. 016118.
[45] M. E. Newman, "Modularity and community structure in networks," *Proc. Nat. Academy Sci. United States America*, vol. 103, no. 23, pp. 8577–8582, 2006.
[46] A. Goldenberg, A. X. Zheng, S. E. Fienberg, and E. M. Airoldi, "A survey of statistical network models," *Found. Trends Mach. Learn.*, vol. 2, no. 2, pp. 129–233, 2010.
[47] N. Alon, E. D. Demaine, M. T. Hajiaghayi, and T. Leighton, "Basic network creation gabmes," *SIAM J. Discr. Math.*, vol. 27, no. 2, pp. 656–668, 2013.
[48] P. Lenzner, "On dynamics in basic network creation games," in *Proc. Int. Symp. Algorithmic Game Theory*, 2011, pp. 254–265.
[49] A. Gulyás, J. Bíró, A. Kőrösi, G. Rétvári, and D. Krioukov, "Complex networks as Nash equilibria of navigation games," arXiv:1412.7229, 2014.
[50] D. Cartwright and F. Harary, "Structural balance: A generalization of Heider's theory," *Psychological Rev.*, vol. 63, no. 5, 1956, Art. no. 277.
[51] F. Heider, "Attitudes and cognitive organization," *J. Psychology*, vol. 21, no. 1, pp. 107–112, 1946.
[52] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropological Res.*, vol. 33, pp. 452–473, 1977.
[53] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.
[54] D. D. Heckathorn and J. Jeffri, "Social networks of jazz musicians," *Changing Beat: A Study Worklife Jazz Musicians*, vol. 3, pp. 48–61, 2003.
[55] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, 2006, Art. no. 036104.
[56] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1729–1736.
[57] J. Hofbauer and W. H. Sandholm, "On the global convergence of stochastic fictitious play," *Econometrica*, vol. 70, no. 6, pp. 2265–2294, 2002.
[58] S. Ehsani, M. Fazli, A. Mehrabian, S. S. Sadeghabad, M. Safari, M. Saghafian, and S. ShokatFadaee, "On a bounded budget network creation game," in *Proc. 23rd Annu. ACM Symp. Parallelism Algorithms Archit.*, 2011, pp. 207–214.
[59] M.-Y. Kao, "Approximate Nash equilibrium," in *Encyclopedia of Algorithms*. Berlin, Germany: Springer, 2008, pp. 46–46.
[60] C. Daskalakis, A. Mehta, and C. Papadimitriou, "Progress in approximate Nash equilibria," in *Proc. 8th ACM Conf. Electron. Commerce*, 2007, pp. 355–358.

MohammadAmin Fazli received the BSc degree in hardware engineering and the MSc and PhD degrees in software engineering from the Sharif University of Technology, in 2009, 2011, and 2015 respectively. He is currently an assistant professor with the Sharif University of Technology and R&D supervisor with Intelligent Information Center resided in this university. His research interests include game theory, combinatorial optimization, computational economics, graphs and combinatorics, complex networks, and dynamical systems.

Abbas Maazallahi received the BSc and MSc degrees in software engineering from the Sharif University of Technology, in 2010 and 2012 respectively. His research interests include social network analysis, big data, and artificial intelligence.

Jafar Habibi is an associate professor in the Department of Computer Engineering, Sharif University of Technology, and the managing director of Intelligent Information Solutions Center. He is a supervisor of Sharif's Robo-Cup Simulation Group and Software Engineering Lab. His research interests include the areas of software engineering, simulation systems, MIS, DSS, and evaluation of computer systems performance.

Moslem Habibi received the BSc and MSc degrees in software engineering from the Sharif University of Technology, in 2010 and 2012 respectively. He is working toward the PhD degree in software engineering at the Sharif University of Technology where his research centers on modeling economic incentives for cloud federation. His research interests include cloud computing, cloud interoperability, cloud computing economic models, and complex networks.

Queries to the Author

Q1. Please provide complete bibliographic details in Refs. [8], [20], [22], [36], [40], and [49].
Q2. Please provide the page range in Ref. [12].