Evolutionary Divide-and-Conquer Algorithm for Virus Spreading Control Over Networks

Tian-Fang Zhao, *Student Member, IEEE*, Wei-Neng Chen, *Senior Member, IEEE*, Sam Kwong, *Fellow, IEEE*, Tian-Long Gu, Hua-Qiang Yuan, Jie Zhang, *Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The control of virus spreading over complex networks with a limited budget has attracted much attention but remains challenging. This article aims at addressing the combinatorial, discrete resource allocation problems (RAPs) in virus spreading control. To meet the challenges of increasing network scales and improve the solving efficiency, an evolutionary divide-and-conquer algorithm is proposed, namely, a coevolutionary algorithm with network-community-based decomposition (NCD-CEA). It is characterized by the community-based dividing technique and cooperative coevolution conquering thought. First, to reduce the time complexity, NCD-CEA divides a network into multiple communities by a modified community detection method such that the most relevant variables in the solution space are clustered together. The problem and the global swarm are subsequently decomposed into subproblems and subswarms with low-dimensional embeddings. Second, to obtain high-quality solutions, an alternative evolutionary approach is designed by promoting the evolution of subswarms and the global swarm, in turn, with subsolutions evaluated by local fitness functions and global solutions evaluated by a global fitness function. Extensive experiments on different networks show that NCD-CEA has a competitive performance in solving RAPs. This article advances toward controlling virus spreading over large-scale networks.**

***Index Terms*—Cooperative coevolution (CC), evolutionary algorithm (EA), networked system, resource allocation, spreading control.**

Manuscript received October 10, 2019; revised December 31, 2019; accepted February 15, 2020. Date of publication March 13, 2020; date of current version June 23, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61976093 and Grant 61876111, in part by the Science and Technology Plan Project of Guangdong Province under Grant 2018B050502006, and in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003. The work of Tian-Fang Zhao and Wei-Neng Chen was supported by the Guangdong–Hong Kong Joint Innovation Platform of Big Data and Computational Intelligence under Grant 2018B050502006. This article was recommended by Associate Editor H. Takagi. *(Corresponding author: Wei-Neng Chen.)*

Tian-Fang Zhao and Wei-Neng Chen are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou 510006, China (e-mail: cwnraul634@aliyun.com).

Sam Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

Tian-Long Gu is with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China.

Hua-Qiang Yuan is with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan 523808, China.

Jie Zhang is with the Computer Science Department, Beijing University of Chemical Technology, Beijing 100029, China.

Jun Zhang is with Hanyang University, Seoul 04763, South Korea.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2020.2975530.

Digital Object Identifier 10.1109/TCYB.2020.2975530

I. Introduction

VIRUS spreading, including the pervasion of infectious diseases and the diffusion of computer malwares, has caused great panic and a huge economic loss in past centuries [1]. Necessary resource interventions have been proved to be effective in virus control. For example, Africa malaria has posed a great threat to the life of people in decades, especially in developing countries, but it has fallen by 40% between 2000 and 2015 due to the wide distribution of preventive resources (insecticide-treated nets) [2]. The recent worldwide cyberattack—WannaCry ransomware attack—has affected more than 150 countries and caused billions of economic losses, and anti-malware programs are the key to ensure the security of networking devices.

Since virus spreading is hard to be simulated in the real world due to huge expenditure and high risk of control failure, mathematical simulation and analog control become effective ways to help determine the public policies in the real-world crises. In virus spreading control, relevant studies can be divided into three complementary lines of research.

A. Topology Adaption

Some early studies based on complex network theory suggested that network topologies took significant impacts on spreading dynamics [3], [4]. As a result, a fair number of topology adaption strategies were developed, for example, removing certain connections by using topology-manipulative algorithms [3], [5] removing a fraction of nodes with high-degree centrality [5], [6] and isolating the nodes at high infection risk [7]–[10]. These strategies could efficiently eradicate virus diffusion to some extent, but they usually ignored the cost of adaption [3], [6], [8] or caused a great loss on network connectivity [3], [5]–[7].

B. Node Intervention

In contrast to the topology adaption, the node intervention emphasized transitioning the node state or reducing the infection rates instead of cutting off contacts in networks.

The measures to prevent epidemic spread includes vaccinating the healthy nodes by the zero-determinant strategy [11] or convex-based optimization [12]; protecting nodes from infection by distributing timely information [13], [14]; and healing the top-q infected nodes by contact tracing [15] or priority planning [16]. The measures to defense malware diffusion includes deploying malware detectors by genetic algorithms [17] and genetic programming [18], and developing anti-malware software by evolutionary computation techniques [19]. Besides, some studies focused on auditing anti-malware tools through the evolving android malware [20], [21], which provided new insights for malware defense. Though the above strategies have shown their advantages in the allocation of single-category resources, it is still a difficulty to integrate the allocation of these different resources.

C. Combinatorial Resource Allocation

Recent studies have paved the way for the allocation of two or more resource categories [22]–[23]. Convex or quasiconvex optimization strategies, such as geometric programming [22], [24], [25] and gradient-based optimization [23], [26], became very popular. These strategies performed well on the combinatorial allocation of continuous resources. But empirically, the real-world resources in virus spreading control are some concrete goods, tools, or services, which have the discrete attribute. Allocating such discrete resources becomes typical subset selection optimization, which is nondeterministic polynomial hard (NP-hard) [27]. The solutions to NP-hard problems cannot be obtained in polynomial time. Therefore, approximate optimization strategies, instead of convex or quasiconvex optimization, become the most promising choice [7]. Besides, since nonlinear objectives in virus spreading control have high computing complexity, most existing strategies were investigated in small-scale networks, such as the networks with $N = 20$ nodes [22], $N = 50$ nodes [23], or $N = 100$ nodes [1], [24], [26]. To conquer the challenge of large-scale networks, divide-and-conquer policies should be considered, which can effectively reduce the dimensions of the problems.

Therefore, the motivation of this article is to design an approximate optimization strategy with a graph-based divide-and-conquer policy, to solve the combinatorial, discrete resource allocation problems (RAPs) in large-scale networks.

As an efficient approximation optimization policy, evolutionary algorithms (EAs) have demonstrated their advantages in solving traditional scheduling problems, such as the knapsack problems [28], cloud resource scheduling [29], task allocation [30], etc. But so far, there are still few cases of EAs for the scheduling of discrete resources in the network-based virus control. One challenge lies in that existing EAs are designed for continuous optimization problems which may be trapped in a local optima in dealing with the discrete optimization problems [28], [31]. Another challenge is that the fitness functions of virus spreading problems are complicated, nonlinear, and high dimensional. As the network scale increases linearly, the dimensionality of the solution space increases exponentially. In large-scale networks with more than 1000 nodes (large-scale optimization), existing EAs lose their effectiveness and efficiency, and thus there is an urgent need to develop novel EAs suitable for such situations [32], [33].

The major contribution of this article is to propose a coevolutionary algorithm with network-community-based decomposition (NCD-CEA) so that the aforementioned issues can be alleviated. The major highlights are as follows.

1) A network-community-based decomposition strategy is designed to help the divide-and-conquer problem. In the strategy, the community structure of a network is detected by a modified Louvain algorithm (MLA). Compared to the original Louvain algorithm (Louvain) [34], an MLA includes a specific control mechanism to control the number of communities and the size of each community, so that super communities and the mini-communities can be avoided. Then, based on the neighborhood propagation features of the problem and community structure characteristics of the network, the strategy divides the problem into multiple subproblems with low-dimensional solution space. In contrast to the existing spreading control policies, the proposed decomposition strategy can effectively reduce the time complexity and improve solving efficiency.
2) An alternative evolution process is designed to coordinate the solving of subproblems and the global problem, corresponding to the local evolution of subswarms and the global evolution of the swarm in coevolutionary algorithms (CEAs). But in existing CEAs, the local fitness evaluations are the same as the global fitness evaluation. If there are intractable objectives in RAPs, the global fitness evaluation becomes especially time consuming. In NCD-CEA, the local fitness functions only refer to the subproblems which have low-dimensional solution space, thereby reducing the execution time of the algorithm. Besides, to avoid trapping into the local optima, we start the evolution of the global swarm after certain intervals to correct and guide the evolution of subswarms. The alternative evolution way can enhance the searching diversity of subswarms and promote global exploration.
3) The proposed NCD-CEA launches a new attempt for solving the combinatorial, discrete RAPs in large-scale networks. As far as we know, this is the first attempt for solving virus spreading control problems by CEAs. Extensive experiments on various complex networks show that NCD-CEA produces high-quality solutions to the problems than its competitors.

The remainder of this article is organized as follows. Section II introduces preliminary knowledge about the cooperative coevolution (CC) approach and community structure detection. Section III formulates the problems. Section IV elaborates on the proposed NCD-CEA. Section V presents experimental results and discussions. Section VI concludes this article.

II. Preliminary

In this section, we provide the background of the CC approach and community structure detection.

A. Cooperative Coevolution Approach

The CC approach provides a user-specified decomposition possibility to divide the population into multiple interacting subpopulations [35], resulting in cooperative coevolution algorithms (CCEAs). CCEAs take advantage of maintaining the solution diversity due to the parallel evolution of multiple subpopulations. For a full review of CCEAs, refer to [36]. We introduce the most relevant studies in the following.

The original CC was used to improve the performance of genetic algorithms (GAs), resulting in cooperative coevolution GAs (CCGAs) [35]. The CCGA was proven to be with lower computational costs than the original GA in solving function optimization problems [35]. Then, Van Den Bergh and Engelbrecht used the CC approach to improve the particle swarm optimizers (PSOs) in 2004 [37], and called the new algorithm as CPSO-S_K. The experimental results showed CPSO-S_K performed better than the original PSOs in increasing the solution diversity. Later, Li and Yao [38] developed a new cooperative coevolving particle swarm optimization (CCPSO2) to solve the high-dimensional, nonseparable, and many-variable problems. In recent years, an increasing number of CCEAs was developed to solve the large-scale optimization and acquired favorable effect [39]. These studies demonstrated that CCEAs had potential in large-scale optimization. Therefore, we consider CCEAs as a good choice for controlling large-scale virus spreading.

Though CCEAs have been receiving much attention for the large-scale optimization, there are still some important challenges unsolved. The first is the problem decomposition and variable linkage learning. Existing nongraph-based CCEAs tend to calculate each element in the solution space independently [35] or calculate the element-cluster gathered by random grouping methods [38] and differential grouping methods [40]. These CCEAs have been tested to be effective and efficient in CEC benchmark sets, but they lost their effectiveness and efficiency in solving network-based optimization problems. That is because variables in network-based optimization problems are not independent of each other but influenced by their neighboring variables, with the neighboring relationships closely connected to network topologies. The second challenge lies in the cooperation way among subpopulations and the fitness evaluation way. After dividing the solution space, how to effectively coordinate multiple interacting subswarms so that the high-quality solutions can be obtained becomes the focus [38]. However, existing fitness evaluation ways take high time and space expenditure for maintaining frequent cooperation, especially in solving large-scale optimization problems. To address the above two challenges, we explore a novel swarm decomposition method and a new fitness evaluation way in this article which is enlightened by the technique of community structure detection.

B. Community Structure Detection

Community structure is very common in the real-world networks, such as population contact networks, social networks, and Internet [41]. It is manifested in dense connections among nodes in same communities and sparse connections among nodes in different communities [42]. Community structure characterizes the topology structure of the network and helps understand how the substructures affect each other [43]. So far, Newman–Girvan's modularity [44] is one of the most popular criteria in the community structure detection, which evaluates how well the network is divided into communities. Concretely, the modularity function, marked as Q, is defined as the difference between the number of edges within communities and the expected number of randomly distributed edges between communities. Given an undirected and unweighted network with N nodes (identified by their number) and m edges, which can be divided into L communities, let C_i represent the ith community, $i = 1, \ldots, L$. The modularity is calculated by

$$Q = \sum_{i=1}^{L} \left(\frac{E_i}{m} - \left(\frac{K_i}{2m} \right)^2 \right) \tag{1}$$

where E_i represents the number of edges existing in C_i, and K_i is the sum of degree of nodes inside the community C_i. A larger value of Q corresponds to the better community structures.

By using the modularity function, we test the classical community structure detection algorithms by conducting preliminary experiments on complex networks. The test algorithms include: the clique percolation method (CPM) [45], the expectation–maximization (EM) algorithm [46], the Girvan–Newman (GN) algorithm [47], the Louvain method (Louvain) [34], the Lancichinetti–Fortunato method (LFM) [48], the label propagation algorithm (LPA) [49], and the structural clustering algorithm for networks (SCANs) [50]. Finally, we have found that the Louvain algorithm outperforms other algorithms in maximizing modularity.

The Louvain algorithm is a two-phase algorithm [34]. It starts with an undirected and unweighted network, and is followed by two phases of operation. The first phase is to divide the network into communities by maximizing the gain of the modularity. The second phase is to build a new network by transforming the communities into nodes, and redefining the multiple edges between two communities as a new weighted edge which takes the sum of weights of former edges as the new weight. The two phases will not stop until there is no more gain of modularity (ΔQ_j). ΔQ_j is calculated by using

$$\begin{aligned} \Delta Q_j = & \left[\frac{\sum_{j,\text{in}} + K_{i,j}}{m} - \left(\frac{\sum_{j,\text{tot}} + K_i}{2m} \right)^2 \right] \\ & - \left[\frac{\sum_{j,\text{in}}}{m} - \left(\frac{\sum_{j,\text{tot}}}{2m} \right)^2 - \left(\frac{K_i}{2m} \right)^2 \right] \end{aligned} \tag{2}$$

where $\sum_{j,\text{in}}$ is the sum of weights of edges within the community C_j. $\sum_{j,\text{tot}}$ is the sum of the weights of the weighted edges

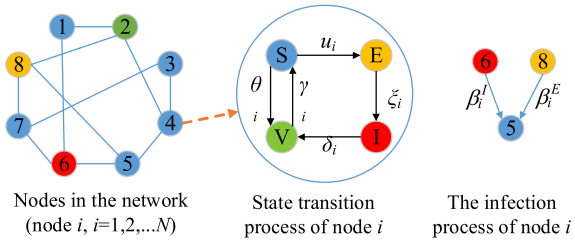

Fig. 1. General view of the SEIV model.

connected to nodes in C_j. $K_{i,j}$ is the sum of the weights of the edges which connect the node i and the nodes in C_j. K_i is the sum of weights of edges linked to node i. m is the sum of the weights of all the edges in the new network. The Louvain algorithm is a stochastic algorithm with a greedy searching strategy, in other words, the results of community structure detection are affected by the sequence of the nodes. In the following, we do not change this greedy strategy in the original Louvain algorithm, but add some extra functions to control the community number and the size of each community.

III. Problem Formulation

In this section, we introduce the models for virus spreading dynamics and resource allocation. Based on the models, two RAPs in virus spreading control are formulated.

A. Virus Spreading Model

The most commonly used spreading models are susceptible-infected (SI) models [51], [52]; SI-susceptible (SIS) models [30]; SI-recovery (SIR) models [14], [53]; and SIR-susceptible (SIRS) models [54]. As a variant of SIRS models, the susceptible–exposed–infected–vigilant (SEIV) model takes advantages on three aspects [22]: 1) it further considers the exposed state E, referring to the asymptomatic state of biological virus infection or the undetectable state of computer virus; 2) it replaces the recovery state R in SIR and SIRS models by the vigilant state V that covers more immune situations such as the information-based immunization; and 3) it can be generalized to SI, SIS, SIR, and SIRS models by adjusting parameters. Therefore, we base this article on the SEIV model.

A general view of the SEIV model is shown in Fig. 1, with N representing the total number of nodes in the network. The capital letters S, E, I, and V represent the susceptible, exposed, infected, and vigilant states, respectively, which are marked with different colors. The states S and V are two healthy states, and the states E and I are two infectious states which can spread the virus. More details about parameters, equations, and optimization objectives in the SEIV model are shown in Table I. By following the equations of state transition process and node infection process in Table I, the spreading dynamics, referring to the state probabilities of all nodes, can be gained. Based on the two optimization objectives of the SEIV model, two optimization problems are formulated in this article.

B. Resource Allocation Model

Generally, control measures to suppress virus spreading correspond to the allocation of specific resources [22]. If we consider biological viruses and computer viruses, respectively, their control measures can be listed as follows. As to biological viruses, control measures include rapid vaccination of uninfected population [12], [55]; isolating the individuals at high infection risks [56]; reducing contacts between individuals [3]; etc. The defense measures of computer viruses are similar to those of biological viruses [57], for instance, allocating detectors to prevent malwares [17], [18]; immunizing nodes [58]; blocking network topology [59]; recovering the attacked systems; enhancing the anti-malware tools [19]–[21]; etc. As a whole, we classify the control resources into three groups: 1) the infection-free resource r_1 (e.g., biological vaccines and system bug fixes), which helps the node i transition from state S to state V by adjusting parameter θ_i; 2) the infection-prevented resource r_2 (e.g., antivirus masks, virus warning messages), which hinders the transition of the node i from state S to state V by adjusting the parameters β_i^E and β_i^I; and 3) the infection-removed resource r_3 (e.g., hospital isolation wards, curative medicines, and antivirus programs), which helps the node i transition from state I to state V by adjusting the parameter δ_i. The allocation model of resources is formulated in Table II.

To simplify the resource allocation process, we assume that when an individual is allocated with some specific resources, its basic attributes (infection rate, recovery rate, immunization rate, etc.) will be consistently changed. For example, when a healthy individual is allocated with the medical drug, it can recover from the infection quickly whenever it is infected. This may cause some resource waste in simulation experiments, but will not affect the practical effects. In reality, individual states are not probabilistic but definite, so the individual state will be binarized according to the corresponding probabilities with detailed operation process shown in [60].

C. Two Optimization Problems

Based on the two optimization objectives in the SEIV model, two RAPs in virus spreading control are formulated as follows.

The RAP with an easy objective (Easy-RAP), which aims to minimize the average infection rate of nodes with limited resource cost, is formulated by

$$\begin{aligned} \min.\ & \overline{u} \\ \text{s.t.}\ & \text{Cost}(\boldsymbol{R}) < C. \end{aligned} \tag{3}$$

The RAP with a hard objective (Hard-RAP), which aims to minimize the maximum real part of eigenvalues of matrix $\boldsymbol{L}'$ with limited resource cost, is formulated by

$$\begin{aligned} \min.\ & \lambda(\boldsymbol{L}') \\ \text{s.t.}\ & \text{Cost}(\boldsymbol{R}) < C. \end{aligned} \tag{4}$$

In both the problems, the optimization vector is the resource allocation matrix $\boldsymbol{R} = [\tau_{ri} | \tau_{ri} \in \{0, 1\}], r = 1, 2, 3, i = 1, \ldots, N$. Each variable τ_{ri} in the matrix represents the allocation situation of the resource r to the node i, as described in Table II. Note that the optimization objectives of the two problems are the fitness functions of EAs in this article and the limited resource budgets are the constraint conditions.

TABLE I
MAIN PARAMETERS AND EQUATIONS OF THE SEIV MODEL

Parameters	Explanation	Values
$G_N=(v,\varepsilon)$	The network with N nodes, where nodes are represented by the set v and edges by the set ε.	$v=\{1, 2, ..., N\}$, $\varepsilon=\{(i,j)\|i,j \in v, i \neq j\}$
$A=[a_{ij}]$	The adjacency matrix of the network G_N.	$a_{ij}=1$ if $(i,j) \in \varepsilon$, otherwise $a_{ij}=0$
θ_i	The immunity acquiring rate of node i transitioning from state S to state V.	$\theta_i \in \{\underline{\theta}_i, \bar{\theta}_i\}$, $(\bar{\theta}_i=0.999, \underline{\theta}_i=0.001)$
β_i^E	The probability of node i is infected by its neighbor in state E.	$\beta_i^E \in \{\underline{\beta}_i^E, \bar{\beta}_i^E\}$, $(\bar{\beta}_i^E=0.5, \underline{\beta}_i^E=0.001)$
β_i^I	The probability of node i is infected by its neighbor in state I.	$\beta_i^I \in \{\underline{\beta}_i^I, \bar{\beta}_i^I\}$, $(\bar{\beta}_i^I=0.3, \underline{\beta}_i^I=0.001)$
ξ_i	The incidence rate of node i transitioning from state E to state I.	$\xi_i \sim N(0.3, 1/6)$, (*constant*)
δ_i	The recovery rate of node i transitioning from state I to state V.	$\delta_i \in \{\underline{\delta}_i, \bar{\delta}_i\}$, $(\bar{\delta}_i=0.999, \underline{\delta}_i=0.01)$
γ_i	The immunity losing rate of node i transitioning from state V to state S.	$\gamma_i \sim N(0.25, 1/6)$, (*constant*)
u_i	The infection rate u_i represents the probability of node i infected by all the neighbors.	$u_i=1-\prod_{j=1}^{N}(1-\beta_i^E a_{ij} p_j^E-\beta_i^I a_{ij} p_j^I)$
Processes	**Symbols**	**Equations**
State transition	$p_i^S, p_i^E, p_i^I, p_i^V$ represents the state probability of node i in S, E, I, V at time t, respectively. $\Delta p_i^S, \Delta p_i^E, \Delta p_i^I, \Delta p_i^V$ represent the state changing probabilities of node i in S, E, I, V, respectively. All the state probability and state changing probabilities are bounded within [0, 1]. At the next time t+1, the state probabilities of nodes are updated by $(p_i^S, p_i^E, p_i^I, p_i^V) \leftarrow (p_i^S, p_i^E, p_i^I, p_i^V)+(\Delta p_i^S, \Delta p_i^E, \Delta p_i^I, \Delta p_i^V)$.	$\begin{cases}\Delta p_i^S=\gamma_i p_i^V-\theta_i p_i^S-(1-\theta_i)u_i p_i^S \\ \Delta p_i^E=(1-\theta_i)u_i p_i^S-\xi_i p_i^E \\ \Delta p_i^I=\xi_i p_i^E-\delta_i p_i^I \\ \Delta p_i^V=\theta_i p_i^S+\delta_i p_i^I-\gamma_i p_i^V\end{cases}$
Stabilization	Let $\boldsymbol{L}'$ represent the linear part of the coefficient matrix of equations about Δp_i^E and Δp_i^I. The maximum real part of eigenvalues of $\boldsymbol{L}'$ is inversely proportional to the speed of stabilization, whose derivation can be learned from [22]. *diag*(·) is a function that fills some variables into the diagonal line of a square matrix, with dimensionality of the matrix equivalent to the number of variables.	$\boldsymbol{L}'=\begin{bmatrix}(\boldsymbol{1}-\boldsymbol{T})\boldsymbol{B}^{\boldsymbol{E}}\boldsymbol{A}-\boldsymbol{F} & (\boldsymbol{1}-\boldsymbol{T})\boldsymbol{B}^{\boldsymbol{I}}\boldsymbol{A} \\ \boldsymbol{F} & -\boldsymbol{D}\end{bmatrix}$ $\boldsymbol{A}=[a_{ij}], \boldsymbol{B}^{\boldsymbol{E}}=diag(\beta_i^E), \boldsymbol{B}^{\boldsymbol{I}}=diag(\beta_i^I),$ $\boldsymbol{T}=diag(\theta_i), \boldsymbol{F}=diag(\xi_i), \boldsymbol{D}=diag(\delta_i)$
Optimization objectives	**Explanation**	**Expressions**
$\bar{u}$	The average infection rate of all nodes, which is ***an easy objective*** with a solving time complexity $O(N)$.	$\bar{u}=\sum_{i=1}^{N} u_i$
$\lambda(\boldsymbol{L}')$	The exponential rate of infectious states (referring to state E and state I) converge to the equilibrium point, which is ***a hard objective*** with a solving time complexity $O(N^2)$.	$\lambda(\boldsymbol{L}')$ is the maximum part of the eigenvalues of $\boldsymbol{L}'$

TABLE II
MAIN PARAMETERS AND FUNCTIONS OF THE RESOURCE ALLOCATION MODEL

Parameters	Explanation	Values
$\boldsymbol{R}$	The $3\times N$ dimensional matrix of resource allocation, with 3 representing the number of resource categories and N representing the number of nodes. $\tau_{ri}=1$ if resource r is allocated to the node i, otherwise $\tau_{ri}=0$.	$\boldsymbol{R}=\begin{bmatrix}\tau_{11} & \tau_{12} & ... & \tau_{1N} \\ \tau_{21} & \tau_{22} & ... & \tau_{2N} \\ \tau_{31} & \tau_{32} & ... & \tau_{3N}\end{bmatrix}$
$Cost(\boldsymbol{R})$	The cost of all the allocated resources, which is a sum of the single-category-resource costs of r_1, r_2, and r_3. To simplify the model, the single-category-resource costs are calculated by the linear functions, which refer to the unit price of the resources and the individual state probabilities.	$\begin{cases}Cost(r_1)=c_1\sum_{i=1}^{N}\tau_{1i}, \ (default: c_1=0.5) \\ Cost(r_2)=c_2\sum_{i=1}^{N}\tau_{2i}, \ (default: c_2=0.5) \\ Cost(r_3)=c_3\sum_{i=1}^{N}\tau_{3i}, \ (default: c_3=0.5)\end{cases}$ $Cost(\boldsymbol{R})=Cost(r_1)+Cost(r_2)+Cost(r_3)$
$Effect(\boldsymbol{R})$	The effect of each allocated resource, which is inclusive of all the effect of each allocated resources. Concretely, (i) The effect of resource r_1 is to vaccinate the susceptible nodes, referring to the immunity acquiring rate θ_i; (ii) The effect of resource r_2 is to protect the susceptible from infection, referring to the infection rate u_i with respect to β_i^E and β_i^I; (iii) The effect of resource r_3 is to cure the infected nodes, in relation to the recovery rate γ_i.	$for\ r_1: \ \theta_i \leftarrow \theta_i+(\bar{\theta}_i-\theta_i)\tau_{1i}$ $for\ r_2: \begin{cases}\beta_i^E \leftarrow \beta_i^E(1-\tau_{2i})+\underline{\beta}_i^E\tau_{2i} \\ \beta_i^I \leftarrow \beta_i^I(1-\tau_{2i})+\underline{\beta}_i^I\tau_{2i}\end{cases}$ $for\ r_3: \delta_i \leftarrow \delta_i+(\bar{\delta}_i-\delta_i)\tau_{3i}$ $(default: \ \theta_i=\underline{\theta}_i, \ \beta_i^E=\bar{\beta}_i^E, \ \beta_i^I=\bar{\beta}_i^I, \ \delta_i=\underline{\delta}_i)$

The solving procedure is as follows. A virus spread environment is first built based on a network topology and the SEIV model is shown in Table I. Then, based on the qualified solutions provided by EAs, the resources are allocated to the nodes accordingly. These resources take effect by the function Effect($\boldsymbol{R}$) in Table II, namely, changing the parameters of the SEIV model. Finally, the fitness values are calculated by averaging over the infection rate of each node (for Easy-RAP) or taking the maximum real part of eigenvalues of the matrix $\boldsymbol{L}'$ (for Hard-RAP).

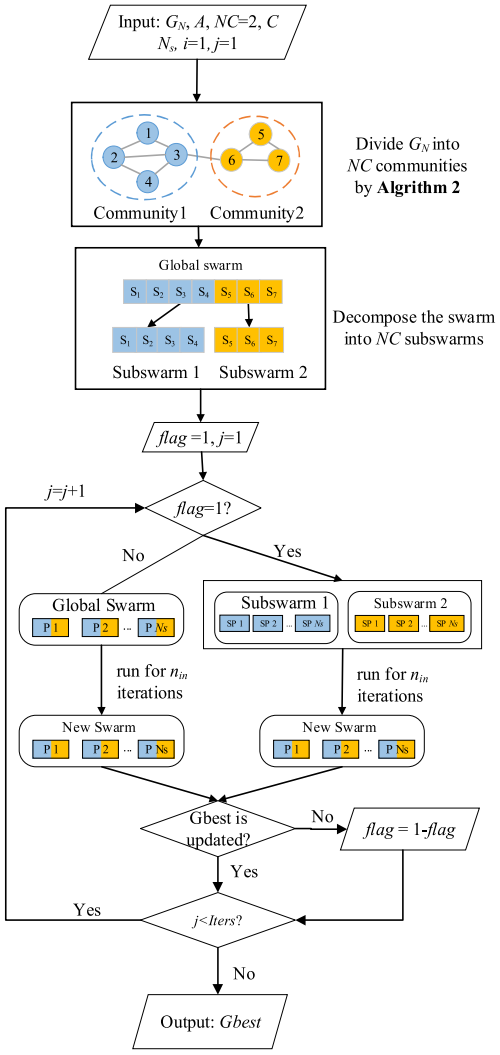

Fig. 2. Framework of NCD-CEA, taking $NC = 2$ as an example.

In summary, the optimization problems combine the resource allocation model and the SEIV model. The total resource cost is related to the state probabilities in virus spreading dynamics, and the resource effect functions are embodied in the two objectives of the optimization problems.

IV. Algorithm Framework

In this section, the proposed NCD-CEA is introduced. It can be used to solve not only virus spreading control problems but also other network-based optimization problems.

A. General Framework

From a systematic perspective, NCD-CEA is an evolutionary divide-and-conquer control policy for virus control. From the perspective of algorithm design, it is a general framework, in which the basic optimizer can be customized. In our preliminary experiments, we find that a majority-voting binary PSO (termed as MVBPSO) [61] can produce the best solutions. Besides, MVBPSO is simply implemented and is very robust to the change of resource cost. Therefore, MVBPSO is selected as the basic optimizer in NCD-CEA.

Algorithm 1 NCD-CEA

Algorithm parameters: swarm size N_s, parameter n_{ic}, local iteration times n_{in}, number of communities NC,
Problem parameters: the network G_N, the cost constraint C.
Output: ***Gbest***.
Initialization: initialize some global solutions for the swarm and qualify them by a **repair mechanism**.
Procedure:

```
The First Step(decomposing):
    Divide the network into NC subnetworks by Algorithm 2.
    Decompose the swarm into NC subswarms.
    Decompose the problem into NC subproblems.
The Second Step (conquering):
  repeat // Iters
    flag = 1
    repeat // n_in
      if flag = 1: // the evolution of subswarms
        Update subswarms by MVBPSO.
        Generate global solutions by merging subsolutions.
      else: // the evolution of the swarm
        Update the swarm by MVBPSO.
      end if
      Repair the unqualified solutions by repair mechanism.
      Update Lbest and Gbest in the swarm.
      if Gbest is not updated:
        flag ← 1 − flag
      end if
    until n_in times.
  until the terminal condition is satisfied.
return Gbest.
```

The major principle of MVBPSO is introduced as follows. Given a swarm with N_s particles, for the kth particle, its position is represented by $\boldsymbol{X}_k = [x_k^{(r,c)}]$, with $r = 1, 2, 3, c = 1, 2, \ldots, N$. $\boldsymbol{Pbest_k} = [pbest_k^{(r,c)}]$ denotes its local best position. $\boldsymbol{Gbest} = [gbest^{(r,c)}]$ represents the global best solution of the whole swarm. Then, the position $\boldsymbol{X}_k$ is updated according to the majority voting of two variants of $\boldsymbol{Pbest_k}$ and $\boldsymbol{Gbest}$. These variants can effectively avoid early maturing, which are generated as follows. We will sample n random elements from $\boldsymbol{Gbest}$ and $\boldsymbol{Pbest_k}$, and store their indices (r, c) into $\boldsymbol{V_k^G}$ and $\boldsymbol{V_k^P}$, respectively. The updating rules of MVBPSO are formulated by

$$\begin{cases} \boldsymbol{V_k^G} = \text{Sampling}(\boldsymbol{Gbest}, \lfloor \text{dis}(\boldsymbol{Gbest}, \boldsymbol{X}_k) \rfloor) \\ \boldsymbol{V_k^P} = \text{Sampling}(\boldsymbol{Pbest_k}, \lfloor \text{dis}(\boldsymbol{Pbest_k}, \boldsymbol{X}_k) \rfloor) \\ \begin{cases} xg^{(r,c)} = 1 - gbest^{(r,c)}, & \text{if } (r,c) \in \boldsymbol{V_k^G} \\ xp_k^{(r,c)} = 1 - pbest_k^{(r,c)}, & \text{if } (r,c) \in \boldsymbol{V_k^P} \\ x_k^{(r,c)} = \begin{cases} xg_k^{(r,c)}, & \text{if } xg_k^{(r,c)} = xp_k^{(r,c)} \\ \begin{cases} 1, & \text{random}(0,1) < 0.5 \\ 0, & \text{otherwise} \end{cases}, & \text{otherwise} \end{cases} \end{cases} \end{cases} \tag{5}$$

where random(0, 1) generates a random number within the range of (0, 1). Sampling($\boldsymbol{Gbest}, n$) is the sampling function, where n is obtained by rounding down the Euclidean distance between $\boldsymbol{Gbest}$ and $\boldsymbol{X}_k$, namely, $n = \lfloor \text{dis}(\boldsymbol{Gbest}, \boldsymbol{X}_k) \rfloor$. $\boldsymbol{XP}_k = [xp_k^{(r,c)}]$ and $\boldsymbol{XG} = [xg^{(r,c)}]$ are the variants of $\boldsymbol{Pbest_k}$ and $\boldsymbol{Gbest}$, respectively.

The framework of NCD-CEA is shown in Fig. 2, in which the parameters of NCD-CEA are defined as follows. *flag* is a self-adaptive parameter which decides the alternating of subswarm evolution or swarm evolution. The parameter NC denotes the number of subnetworks/communities (also the number of subswarms). The parameter n_{in} represents the

Algorithm 2 MLA

Input: the network G_N, the adjacency matrix $\boldsymbol{A}$, the number of communities *NC*.
Output: communities
Divide G_N into communities by **the Louvain algorithm**.
Aggregation Function:
  repeat
    The first phase (aggregate the two smallest community-nodes):
      Select the two community-nodes with smallest size.
      Merge the two community-nodes to build a new community.
      The second phase (build a new community-node):
        Transform the community found in the first phase into a new community-node, whose size is the number of nodes in the community, and whose weight is the sum of the weights of edges inside the community.
        The new edge denotes the links between communities, whose weight is the sum of the weights of edges between nodes in the two communities.
  until the number of communities is not larger than *NC*.
Dichotomy Function:
  repeat
    Select the community of largest size.
    Build a new graph based on the largest community.
    Divide the graph into communities by **the Louvain algorithm**.
    Merge the communities by **Aggregation Function** until there are 2 communities in the graph.
  until the number of communities is not smaller than *NC*
return the independent communities generated from G_N

iteration times of each *flag* turn. *Iters* defines the terminal condition of the algorithm, that is, the execution of the algorithm is terminated when the algorithm runs for *Iters* iterations. The pseudocode of NCD-CEA is shown in Algorithm 1, There are three main processes.

1) *Initialization:* First, a population of solutions are randomly generated within the scope of $\{0, 1\}^{(3,N)}$. If a randomly generated solution fails to satisfy the constraint, a repair mechanism will be applied to generate qualified solutions. The repair mechanism works by repeatedly removing some resources from an unqualified solution, namely, changing the corresponding bits from 1 to 0, until the constraint is satisfied. To accelerate the repairing process, we randomly remove two resources at each repairing step.
2) *Decomposition:* The network is decomposed into *NC* communities by Algorithm 2. Then, the global swarm can be decomposed to *NC* subswarms according to the mapping between the network and communities. Each subswarm solves an independent subproblem, which is extracted from the global problem. To achieve this, we first build an independent subnetwork from the community. Then, the local fitness functions are built on the subnetwork, whose calculation is the same as the global fitness function, and the constraint cost *C* is decomposed into *NC* parts according to the ratios between nodes in subnetworks and nodes in the network. Due to the good locality of the problem, such kind of decomposition is feasible. More details about the decomposition processes are introduced in Section IV-B.
3) *Conquering:* During the evolution of subswarms, each subswarm generates a subsolution to the corresponding subproblem. The *NC* subsolutions generated by different subswarms form a complete solution to the entire problem. In contrast, the evolution of the global swarm directly generates a complete solution to the problem, but it takes a much longer time than the evolution of subswarms. Similar to the initialization process, a complete solution may not satisfy the constraint, thus we also need to use the repair mechanism to make the complete solution become feasible. In the proposed NCD-CEA, the evolution of subswarms and that of the global swarm work cooperatively, which will be introduced in detail in Section IV-C.

B. Network-Community-Based Decomposition

An MLA is presented in Algorithm 2 to divide the network into communities. Compared to the original Louvain algorithm, MLA includes two additional control mechanisms:

1) *The Community Number Control:* The number of communities in the original Louvain is uncertain due to the greedy strategy. Too many communities in the network decrease the influence of subsolutions, while too few communities increase the computational cost. Therefore, effective control to the number of communities is required.
2) *The Community Size Control:* The original Louvain algorithm may generate communities with very uneven sizes. The super communities become the bottleneck of the algorithm and slow down the algorithm execution time, while the mini-communities waste the machine resources.

In fact, the community size and the community number depend on each other. Therefore, a specific control mechanism is designed to exert both kinds of control implemented by aggregation function and dichotomy function in Algorithm 2. We use the aggregation function to reduce the number of communities by aggregating the two smallest communities, and dichotomy function to increase the number of communities by dimidiating the super communities by following the principle of modularity maximization.

After the community structure detection, each community is formulated as an independent undirected-and-unweighted subgraph, which serves as background environments of the subproblems. Synchronously, the global swarm is decomposed into multiple subswarms, where each subswarm inherits a proportion of information of the global swarm.

C. Alternative Evolution Process

In our preliminary experiments, we found that the global swarm evolution promotes the exploitation, but the global fitness evaluation is very time consuming especially in large scale networks. Though the fitness evaluation and the evolution of subswarms were time saving, it might be trapped into local optima due to the loss of global evaluations. To balance the exploitation and exploration, an alternative evolution process is designed in NCE-CEA.

Details about the alternative evolution process are embodied in Algorithm 1, which can be described as follows. We use a self-adaptive parameter $flag \in \{0, 1\}$ to determine the specific evolution way in the next generation. If $flag = 1$, the

offspring is generated by the cooperation of subswarms, otherwise it is generated by the evolution of the global swarm. Only when ***Gbest*** is not updated, *flag* is updated to $1-flag$. We set $flag = 1$ by default. Whatever the turn is the subswarms or global swarm, they will evolve for n_{in} times to fully explore the located solution space. Multiple subsolutions provided by the corresponding subswarms will constitute a complete global solution that can be used to update ***Gbest***.

The two evolution processes play different roles in the algorithm: 1) the evolution of subswarms enhances the local searching, for the variables in each subswarm are closely connected to each other, which facilitate the local fitness evaluations and 2) the evolution of the global swarm promotes the global searching and improves the solution diversity by learning from its own best experience and the entire swarm's best experience. Therefore, to some extent, the evolution of subswarms acts as the early accelerator and the evolution of the global swarm acts as an engine of jumping out of local optima.

V. Experiments

In this section, we first provide the parameter configuration used in the experiments. Then, we verify the competitive performance of NCD-CEA by comparing it with different variants of BPSO and some state-of-the-art EAs. The parameter sensitivity analysis is provided later. Finally, the model stability analysis shows that NCD-CEA can effectively prevent virus spread.

A. Experimental Configuration

Network Configuration: Detailed information about the networks is shown in Table SI in the supplementary material, including nine networks with different characteristics and size. All the artificial networks, respectively, regular (RG) networks [60], Barabasi–Albert (BA) scale-free networks [62], and Watts–Strogatz (WS) small world networks [63], are generated by the Python-Networkx package under the environment of Anaconda 3. The degree of nodes in RG networks is fixed, thereby RG networks have ordered structure but poor scalability. The degree of nodes in BA networks presents power-law distributions due to the preferential attachment, thereby BA networks are robust to random failures but weak to deliberate attacks due to such scale-free property [62]. WS networks are generated by first initializing a neighbor coupled network with fixed node degree, and then randomly rewiring each edge by a probability p. WS networks are characterized by RG networks and random networks. The real networks are two sets of human contact network data and a computer network data, namely, Primary School Temporal Network Data in BMC Infectious Diseases 2014 (*Ps-contact*)[1] [64], Face-to-Face Behavior Data of People during the Exhibition Infectious in 2009 at the Science Gallery in Dublin (*Ex-contact*)[2] [65], and DNC emails corecipients network in the 2016 Democratic National Committee email leak (*Email*)[3] [66].

Problems Configuration: Before allocating the resources, we first build a virtual virus spread scenario by evolving the SEIV model for several times with the time interval called as resource intervention time t. Originally, two fixed nodes act as the source of infection (state E). At time t, there has been a proportion of nodes are infected, shown by $p^E + p^I$. In the right part of Table SI in the supplementary material, we provide the optimal solutions for Hard-EAP and Easy-EAP based on the assumption of infinite resource budget, namely, the lower bound of $\overline{u}$ and lower bound of λ. The values of the optimal solutions differ in the network environment and spread dynamically. The lower bound of λ is very close but not equal to -0.01. In reality, the resource budget cannot be infinite, so we set it as $C = 0.3C_{\max}$ in the following experiments, where $C_{\max}$ representing the upper bound of resource cost when resources r_1, r_2, and r_3 are allocated to all nodes in the network.

Comparison Algorithms: The proposed algorithm is compared with multiple versions of BPSO [67] and some state-of-the-art EAs. The parameters of all algorithms used in the comparison have been carefully adjusted to the best. The detailed description and parameter configurations for the comparison algorithms are shown in Table SII in the supplementary material. The BPSO variants include BPSOS2, BPSOS3, BPSOS4, BPSOV1, BPSOV2, and BPSOV3. Thereinto, the classical particle swarm optimization (BPSO or BPSOS1) can be represented by

$$\begin{cases} v_k^{(r,c)} \leftarrow wv_k^{(r,c)} + c_1 r_1 (pbest_k^{(r,c)} - x_k^{(r,c)}) \\ \qquad\qquad + c_2 r_2 (gbest^{(r,c)} - x_k^{(r,c)}) \\ S1(v_k^{(r,c)}) = 1 \Big/ (1 + \exp(-v_k^{(r,c)})) \\ x_k^{(r,c)} = \begin{cases} 1, & \text{if } rand(0,1) < S1(v_k^{(r,c)}) \\ 0, & \text{otherwise} \end{cases} \end{cases} \tag{6}$$

where w is the inertia coefficient, c_1 and c_2 are two acceleration coefficients, and r_1 and r_2 are two random parameters uniformly distributed within [0, 1]. The discretization method is in BPSO is a standard sigmoid function, called as "S1(·)." Besides, there are still other S-shape and V-shape discretization methods can replace "S1(·)," which are listed as follows:

$$\begin{cases} S2(v_k^{(r,c)}) = 1 \Big/ (1 + \exp(-2v_k^{(r,c)})) \\ S3(v_k^{(r,c)}) = 1 \Big/ (1 + \exp(-v_k^{(r,c)}/2)) \\ S4(x_k^{(r,c)}, v_k^{(r,c)}) \leftarrow 1 \Big/ (1 + \exp(-x_k^{(r,c)} - \left\lfloor v_k^{(r,c)} \right\rfloor)) \\ V1(v_k^{(r,c)}) = \begin{cases} 1 - 2 \Big/ (1 + \exp(-v_k^{(r,c)})), & \text{if } v_k^{(r,c)} \le 0 \\ 2 \Big/ (1 + \exp(-v_k^{(r,c)})) - 1, & \text{otherwise} \end{cases} \\ V2(v_k^{(r,c)}) = \left| \frac{2}{\pi} ac\tan(\frac{\pi}{2} v_k^{(r,c)}) \right| \\ V3(v_k^{(r,c)}) = \left| \tanh(v_k^{(r,c)}) \right| \end{cases} \tag{7}$$

[1]The Ps-contact network data can be downloaded from: http://www.sociopatterns.org/datasets/primary-school-temporal-network-data/.

[2]The Ex-contact network data can be downloaded from: http://konect.uni-koblenz.de/networks/sociopatterns-infectious.

[3]The Email network data can be downloaded from: http://konect.uni-koblenz.de/networks/dnc-corecipient.

Table SIII in the supplementary material shows that MVBPSO is the best algorithm in above BPSO variants, so we only take MVBPSO as the comparison algorithm in the following. Besides, some popular EAs and some state-of-the-art algorithms are compared. Two GAs are included: 1) the GAs with elitist expected value selection, partially matched crossover, and bit-reverse mutation (called as GA1) [68] and 2) the GA with tournament selection, basic crossover rate *pc*, and basic mutation rate *pm* (called as GA2) [69]. The above GAs have been successfully used in solving malware prevention problems [19], [20]. Some state-of-the-art EAs include: ant colony system for solving 0/1 knapsack problems (ACS) [70], self-adaptive differential evolution with neighborhood search (SaNSDE) [71], the competitive swarm optimizer (CSO) [72], and the level-based learning swarm optimizer (LLSO) [32]. Besides, the comprehensive learning PSO (CLPSO) [73] proposed by Liang *et al.* is also investigated.

Algorithm Configuration: The NCD-CEA and the algorithms used in comparison share the same initialization and solution repairing methods. The final solutions are obtained by averaging over some independent runs of algorithms called as *Runs*. In each run, their basic settings also stay the same. For simplicity, we use a fixed swarm size $Ns = 20$ for all experiments. In small-scale networks (RG100, BA100, and WS100), we set $Iters = 500$ and $Runs = 50$. In larger-scale networks (WS500, WS1000, Ps-contact, Ex-contact, and Email), we set $Iters = 1000$ and $Runs = 30$.

In addition, NCD-CEA and the comparison algorithms are implemented by Python language. All the experiments are executed on a PC with 8 Intel Core i5-3470 3.20-GHz CPUs, 8-Gb memory. The software platform is Anaconda3 for Ubuntu 12.04 LTS 64-b system.

B. Parameter Analysis

In NCD-CEA, two parameters are introduced: 1) the number of subswarms (equal to the number of communities *NC*) in the strategy of *network-community-based decomposition* and 2) the local iteration times (n_{in}) in the *alternative evolution process*. NCD-CEA with higher value of *NC* can be used to deal with higher dimensional optimization problems for the parallel evolution of the subswarms that contributes a lot to the speedup of computational time in NCD-CEA. But due to the limitations of computational resources, we set the maximum value of *NC* only to 8 in this article. Finally, we set $NC \in \{1, 2, 4, 8\}$ and $n_{\text{in}} \in \{2,\ 5,\ 10, 20\}$ to test the algorithm performance.

The experimental results are shown in Tables SIV and SV in the supplementary material. The best parameter settings depend on the network topologies and differ on the optimization objectives. Overall, for solving Easy-EAP, NCD-CEA is not sensitive to parameter *NC* and a large value of parameter n_{in} ($n_{\text{in}} = 10,\ n_{\text{in}} = 20$) can produce better solutions as the network scale increases. For solving Hard-EAP, the higher values of n_{in} contribute to better solutions (see Fig. 3). The influence of *NC* depends on network topology. In small-scale networks ($N = 100$), the original MVBPSO (*equal to* NCD-CEA with $NC = 1$) is mostly recommended, because the local evolution of subswarms contribute less to the global optimization. While in networks with a larger scale ($N > 100$), there is a wider solution space, and the global evolution cannot locate the better solution quickly. In this situation, the local search conducted by the separable subswarm can accelerate the updating process. Meanwhile, since the subsolutions are evaluated by local fitness functions which consume less time than the global fitness evaluation, especially when the objective calculation is computationally complex.

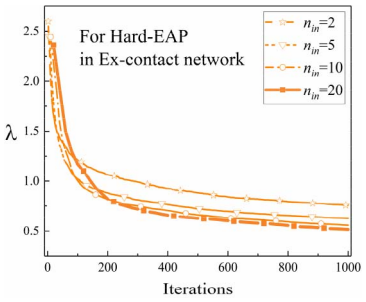

Fig. 3. Convergence speed of NCD-CEA with different n_{in}.

It is worth mentioning that a fixed decomposition operation in NCD-CEA is better than the dynamic and repetitive network decomposition, according to our preliminary experimental results. Therefore, we keep the initially generated communities unchanged in the later stage. Moreover, for the networks with large scale, NCD-CEA with a large *NC* will further reduce the time complexity due to the lower cost of fitness evaluation, and it may produce better solutions, for example, solutions referring to NCD-CEA with $NC = 8$ in Ps-contact and Ex-contact networks are better than the those about $NC = 4$ (see Table SV). But to verify the competitiveness of NCD-CEA, we set $NC = 4$ for solving the two problems in all networks.

C. Comparison Experiments

In this section, we compare NCD-CEA with different EAs. Table III shows the comparison results among NCD-CEA and eight popular EAs for solving Easy-EAP and Hard-EAP in networks with $N < 500$. Thereinto, we find the top-three algorithms are NCD-CEA, MVBPSO, and CLPSO. Comparison results for the three algorithms are shown in Table IV, with the network size $N \geq 500$.

For effective comparisons, the Kruskal–Wallis nonparametric statistical test [74] is first used for multiple comparisons, followed by pairwise comparisons using the Wilcoxon rank-sum test [75], and the critical values are finally corrected using Holm's method [76]. The null hypothesis (H0) assumes that there is no significant difference between the compared objects. For the Kruskal–Wallis test, if the *p*-value is smaller than 0.05, then the null hypothesis is refused, namely, the algorithms provide significantly different solutions. For the Wilcoxon rank-sum test, we choose the algorithm with the best "mean" value as the control group, and others as the experimental group by turn. When the *p*-value is larger than the corrected critical value, the solution of the experimental group

TABLE III
ALGORITHM COMPARISON ON SMALL-SCALE NETWORKS WITH $N < 500$. (a) FOR EASY-EAP. (b) FOR HARD-EAP.

(a)

Net.	Quality	NCD-CEA	MVBPSO	GA1	GA2	ACS	SaNSDE	CLPSO	CSO	LLSO
RG100	Mean	**6.61e-05**	1.03e-02	6.19e-03	6.08e-03	1.18e-02	2.23e-03	7.36e-05	5.73e-03	2.83e-03
	Best	6.61e-05	1.29e-03	3.77e-04	2.59e-03	8.04e-03	4.28e-04	6.97e-05	2.37e-03	1.19e-03
	Std	1.28e-07	2.09e-02	2.60e-03	8.63e-04	1.62e-03	7.62e-04	1.75e-06	9.12e-04	8.38e-04
	Kruskal Wallis	1.04e-144 (<0.05, Significant difference)								
	Wilcoxon	-	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18
	Holm	-	5.00e-02	2.50e-02	1.67e-02	1.25e-02	1.00e-02	8.33e-03	7.14e-03	6.25e-03
BA100	Mean	**3.06e-02**	3.10e-02	2.20e-01	2.28e-01	2.61e-01	1.10e-01	5.45e-02	2.26e-01	1.96e-01
	Best	2.77e-02	2.77e-02	1.07e-01	2.10e-01	2.34e-01	9.86e-02	4.31e-02	2.08e-01	1.80e-01
	Std	2.17e-03	2.40e-03	4.17e-02	6.05e-03	8.68e-03	4.49e-03	5.48e-03	5.96e-03	7.50e-03
	Kruskal Wallis	7.90e-134 (<0.05, Significant difference)								
	Wilcoxon	-	4.38e-01	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18
	Holm	-	5.00e-02	2.50e-02	1.67e-02	1.25e-02	1.00e-02	8.33e-03	7.14e-03	6.25e-03
WS100	Mean	**9.40e-05**	2.18e-03	1.02e-02	7.80e-03	1.47e-02	3.14e-03	9.37e-05	7.42e-03	3.93e-03
	Best	8.71e-05	-2.07e-03	3.74e-03	5.42e-03	9.75e-03	6.70e-04	8.92e-05	5.33e-03	1.16e-03
	Std	2.07e-05	1.78e-03	2.42e-03	1.03e-03	1.94e-03	9.26e-04	2.16e-06	9.79e-04	9.44e-04
	Kruskal Wallis	3.42e-148 (<0.05, Significant difference)								
	Wilcoxon	-	3.37e-14	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18	6.86e-18
	Holm	-	5.00e-02	2.50e-02	1.67e-02	1.25e-02	1.00e-02	8.33e-03	7.14e-03	6.25e-03
	Average time	***1.24***	*1.66*	*2.32*	*6.76*	*2.52*	*4.33*	*2.98*	*4.20*	*3.90*
Ps-contact	Mean	**1.70e-01**	1.74e-01	6.26e-01	6.33e-01	6.79e-01	3.21e-01	1.89e-01	6.32e-01	5.72e-01
	Best	1.62e-01	1.66e-01	5.04e-01	6.13e-01	6.52e-01	3.03e-01	1.79e-01	6.14e-01	5.48e-01
	Std	4.71e-03	5.11e-03	5.17e-02	6.98e-03	1.21e-02	6.10e-03	4.90e-03	7.11e-03	1.36e-02
	Kruskal Wallis	1.52e-51 (<0.05, Significant difference)								
	Wilcoxon	-	7.10e-04	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11
	Holm	-	5.00e-02	2.50e-02	1.67e-02	1.25e-02	1.00e-02	8.33e-03	7.14e-03	6.25e-03
	Average time	*12.46*	***9.50***	*19.66*	*32.66*	*11.50*	*33.00*	*12.63*	*22.93*	*20.80*
Ex-contact	Mean	**9.79e-02**	1.08e-01	4.59e-01	5.01e-01	5.31e-01	2.38e-01	1.79e-01	4.99e-01	4.60e-01
	Best	8.25e-02	9.43e-02	3.36e-01	4.86e-01	5.01e-01	2.19e-01	1.49e-01	4.90e-01	4.40e-01
	Std	5.68e-03	5.97e-03	5.88e-02	4.63e-03	9.40e-03	4.76e-03	1.04e-02	4.76e-03	8.02e-03
	Kruskal Wallis	1.56e-49 (<0.05, Significant difference)								
	Wilcoxon	-	1.20e-07	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11
	Holm	-	5.00e-02	2.50e-02	1.67e-02	1.25e-02	1.00e-02	8.33e-03	7.14e-03	6.25e-03
	Average time	***21.70***	*25.03*	*47.30*	*77.25*	*27.46*	*68.33*	*28.90*	*48.36*	*43.80*

(b)

Net.	Quality	NCD-CEA	MVBPSO	GA1	GA2	ACS	SaNSDE	CLPSO	CSO	LLSO
RG100	Mean	**1.50e-02**	**8.23e-03**	2.05e-01	2.16e-01	3.03e-01	1.19e-01	1.29e-01	2.22e-01	2.09e-01
	Best	1.21e-03	1.04e-03	-2.48e-03	1.69e-01	2.46e-01	5.51e-03	7.83e-02	1.95e-01	1.69e-01
	Std	2.73e-02	1.55e-02	6.83e-02	1.52e-02	2.42e-02	5.67e-02	2.16e-02	1.32e-02	1.52e-02
	Kruskal Wallis	1.29e-77 (<0.05, Significant difference)								
	Wilcoxon	6.99e-01$^{=}$	-	1.14e-15	6.86e-18	6.86e-18	1.14e-15	2.87e-17	6.86e-18	6.86e-18
	Holm	5.00e-02	-	2.50e-02	1.00e-02	8.33e-03	1.67e-02	1.25e-02	7.14e-03	6.25e-03
BA100	Mean	**-1.92e-05**	2.20e-03	4.00e-01	3.44e-01	7.14e-02	1.71e-01	8.14e-02	3.63e-01	3.02e-01
	Best	-4.36e-03	-1.58e-03	3.98e-02	2.56e-01	-1.03e-03	1.13e-02	5.14e-03	2.78e-01	1.95e-01
	Std	2.43e-03	1.49e-03	1.04e-01	3.23e-02	7.81e-02	2.73e-02	5.07e-02	3.08e-02	3.35e-02
	Kruskal Wallis	7.62e-82 (<0.05, Significant difference)								
	Wilcoxon	-	2.42e-07	6.86e-18	6.86e-18	1.63e-13	6.86e-18	7.73e-18	6.86e-18	6.86e-18
	Holm	-	5.00e-02	1.67e-02	1.25e-02	2.50e-02	1.00e-02	8.33e-03	7.14e-03	6.25e-03
WS100	Mean	2.88e-03	**2.56e-03**	2.36e-01	2.36e-01	8.03e-02	1.07e-01	8.35e-02	2.40e-01	2.28e-01
	Best	-2.10e-03	-1.85e-03	9.22e-03	2.01e-01	-2.35e-03	-5.94e-04	9.00e-03	1.93e-01	1.66e-01
	Std	2.42e-03	2.56e-03	7.60e-02	1.36e-02	8.14e-02	6.35e-02	4.69e-02	1.85e-02	1.81e-02
	Kruskal Wallis	6.72e-73 (<0.05, Significant difference)								
	Wilcoxon	4.91e-01	-	8.21e-18	6.86e-18	6.96e-05	5.38e-15	9.26e-18	6.86e-18	6.86e-18
	Holm	5.00e-02	-	1.25e-02	1.00e-02	2.50e-02	1.67e-02	8.33e-03	7.14e-03	6.25e-03
	Average time	***1.54***	*4.62*	*6.00*	*6.38*	*5.62*	*5.66*	*2.33*	*3.67*	*8.52*
Ps-contact	Mean	**1.04e+00**	1.16e+00	7.99e+00	8.64e+00	4.29e+00	3.53e+00	3.06e+00	8.64e+00	7.37e+00
	Best	7.56e-01	9.12e-01	4.44e+00	8.05e+00	3.55e+00	2.19e+00	2.73e+00	8.17e+00	6.84e+00
	Std	1.40e-01	1.41e-01	1.52e+00	1.96e-01	3.23e-01	4.16e-01	1.24e-01	2.00e-01	2.67e-01
	Kruskal Wallis	1.72e-49 (<0.05, Significant difference)								
	Wilcoxon	-	2.56e-03	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11
	Holm	-	5.00e-02	2.50e-02	1.67e-02	1.25e-02	1.00e-02	8.33e-03	7.14e-03	6.25e-03
	Average time	***13.43***	*18.33*	*34.00*	*69.93*	*38.33*	*49.73*	*19.16*	*33.73*	*41.7*
Ex-contact	Mean	**5.17e-01**	5.41e-01	2.15e+00	2.18e+00	1.20e+00	1.06e+00	1.06e+00	2.20e+00	1.91e+00
	Best	4.57e-01	4.47e-01	5.59e-01	1.92e+00	9.61e-01	7.57e-01	9.38e-01	1.96e+00	1.81e+00
	Std	3.12e-02	4.12e-02	4.39e-01	7.91e-02	1.15e-01	9.37e-02	4.57e-02	7.99e-02	5.98e-02
	Kruskal Wallis	6.84e-48 (<0.05, Significant difference)								
	Wilcoxon	-	2.19e-02	3.18e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11	2.87e-11
	Holm	-	5.00e-02	2.50e-02	1.67e-02	1.25e-02	1.00e-02	8.33e-03	7.14e-03	6.25e-03
	Average time	***50.23***	*59.00*	*113.33*	*182.73*	*118.33*	*156.43*	*63.00*	*92.67*	*107.76*

is regarded as being as good as the one in the control group marked by a superscript "=". As a corrected version of the Bonferroni method, Holm's method is used to generate the corrected critical values. As a corrected version of the Bonferroni method, it is less strict and becomes easier to detect significant differences.

Table IV shows the experimental results in networks with $N < 500$. For Easy-EAP which has a linearly separable

TABLE IV
ALGORITHM COMPARISON ON LARGER-SCALE NETWORKS WITH $N >= 500$

		For Easy-EAP			For Hard-EAP		
Net.	**Quality**	**NCD-CEA**	**MVBPSO**	**CLPSO**	**NCD-CEA**	**MVBPSO**	**CLPSO**
WS500	Mean	**5.17e-02**	8.36e-02	1.42e-01	**4.75e-01**	5.08e-01	7.05e-01
	Best	3.63e-02	7.40e-02	1.30e-01	4.14e-01	4.39e-01	6.66e-01
	Std	8.11e-03	5.19e-03	4.51e-03	3.66e-02	2.48e-02	2.03e-02
	Kruskal Wallis	6.59e-18 (<0.05, Significant difference)			1.17e-14 (<0.05, Significant difference)		
	Wilcoxon	-	2.87e-11	2.87e-11	-	1.03e-03	2.87e-11
	Holm	-	5.00e-02	2.50e-02	-	5.00e-02	2.50e-02
	Average time	***25.96***	*35.53*	*37.13*	***82.67***	*88.34*	*94.00*
Email	Mean	**7.62e-02**	1.14e-01	1.95e-01	**2.27e-01**	3.02e-01	1.91e+00
	Best	6.46e-02	1.08e-01	1.85e-01	1.77e-01	2.11e-01	1.63e+00
	Std	4.91e-03	3.85e-03	5.19e-03	3.52e-02	5.70e-02	1.42e-01
	Kruskal Wallis	6.59e-18 (<0.05, Significant difference)			2.91e-16 (<0.05, Significant difference)		
	Wilcoxon	-	2.87e-11	2.87e-11	-	1.73e-07	2.87e-11
	Holm	-	5.00e-02	2.50e-02	-	5.00e-02	2.50e-02
	Average time	***78.26***	*119.50*	*115.53*	***345.90***	*429.65*	*455.15*
WS1000	Mean	**1.35e-01**	2.79e-01	3.80e-01	**1.07e+00**	1.10e+00	1.55e+00
	Best	1.11e-01	2.71e-01	3.73e-01	9.97e-01	1.02e+00	1.47e+00
	Std	1.21e-02	5.18e-03	4.67e-03	3.97e-02	3.14e-02	2.33e-02
	Kruskal Wallis	6.59e-18 (<0.05, Significant difference)			2.23e-14 (<0.05, Significant difference)		
	Wilcoxon	-	2.87e-11	2.87e-11	-	4.97e-03	2.87e-11
	Holm	-	5.00e-02	2.50e-02	-	5.00e-02	2.50e-02
	Average time	***81.23***	*163.55*	*134.03*	***516.8***	*587.6*	*665.00*

object, NCD-CEA wins the first place according to the results obtained by averaging over multiple independent runs. The p-value obtained by the Kruskal–Wallis test shows that there is a significant difference among all comparison algorithms. The p-values of Wilcoxon rank-sum test demonstrate that NCD-CEA is significantly distinguished from other algorithms. Since NCD-CEA has best values of "mean", "best", and "std" in most networks, its first place in solving Easy-EAP is solid. For Hard-EAP which has a nonlinearly nonseparable objective, NCD-CEA defeats MVBPSO and other algorithms in three of five networks (BA100, Ps-contact, and Ex-contact), and has equal solution quality with MVBPSO in the RG100 network, and lose out to MVBPSO in the WS100 network. For both the problems, NCD-CEA, MVBPSO, and CLPSO-S1 defeat most of the other algorithms and win the first three places. Then, the places are SaNSDE-V1 > LLSO-V3 > CSO-S1 > (GA1, GA2). ACS ranks the last for Easy-EAP, but it defeats LLSO-V3, CSO-S1, GA1, and GA2 in four of five networks for Hard-EAP.

Table IV shows the experimental results in networks with $N \geq 500$. Only the solutions provided by the first three algorithms (NCD-CEA, MVBPSO, and CLPSO) are shown. In these networks with larger size, NCD-CEA gives the best solutions in all the three network and both the problems. Results of Kruskal–Wallis test show the differences among the three algorithms. Results of Wilcoxon rank-sum test show that there is a significant difference between NCD-CEA and other algorithms in solving Easy-EAP and Hard-EAP. In other words, the leading performance of NCD-CEA is not accidental.

The above observations produce a crucial conclusion: NCD-CEA can produce the highest quality solutions in all comparison algorithms. It is naturally suitable for solving the problems with the linearly separable object and large-scale discrete optimization due to its divide-and-conquer strategy.

Convergence Comparisons: The convergence behavior of several representative algorithms in solving Easy-EAP

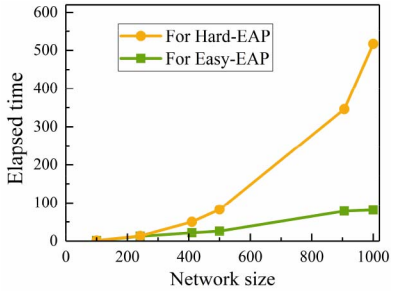

Fig. 4. Time comparison of NCD-CEA for Easy-EAP and Hard-EAP happening in different size of networks.

(Fig. S1) and Hard-EAP (Fig. S2) in the supplementary material. From the algorithm comparison results in Figs. S1(a)–(b) and S2(a)–(b), three conclusions can be drawn: 1) NCD-CEA is able to preserve the highest quality solutions and competitive convergence performance than its competing algorithms. It conserves this superiority and stable performance in most of the networks; 2) MVBPSO and CLSPO are second to NCD-CEA, which have good ability to jump out of local optima, but the convergence speed of them is still lower than NCD-CEA on most networks; and 3) for Hard-EAP, ACS can produce good solutions at early stage due, but it becomes easy to be trapped into local optima at the later stage [see Fig. S2(a) and (b) in the supplementary material]. This phenomenon is caused by the introduction of heuristic information and the utilization of an aggressive pseudorandom proportion selection rule in ACS. This conclusion is consistent with previous studies [77], [78].

In Figs. S1(c)–(e) and S2(c)–(e) in the supplementary material, we depict the convergence trends of the first three algorithms, with the original lines with square points to depict the trends of NCD-CEA. In the network with larger scales, such as WS500, Email, and WS1000 networks, MVBPSO and CLPSO significantly lose their competitiveness.

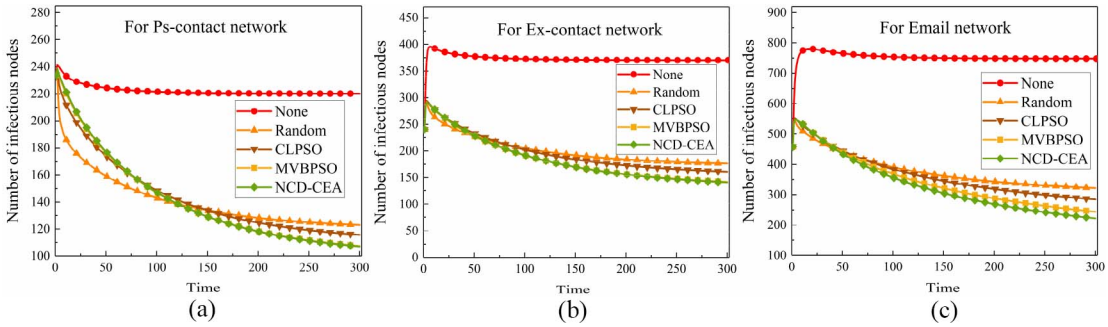

Fig. 5. Number of infectious nodes after resource intervention, taking Hard-EAP as example. (a) Ps-contact with 242 nodes. (b) Ex-contact with 410 nodes. (c) Email with 906 nodes.

Time Comparisons: To validate the competitive efficiency of NCD-CEA, we compare the execution time of NCD-CEA and the comparison algorithms. Results are obtained by averaging over all independent runs, where each execution process is not disturbed by other programs to make sure the execution time is accurate. The platform and programming language keep the same for all comparison algorithms, which have been introduced in Section V-A. All results are shown in "Average time" fields in Tables III and IV. Since the elapsed time of algorithms in RG100, BA100, and WS100 are almost equal, we only showed the average time for WS100, Ps-contact, Ex-contact, WS500, Email, and WS1000 networks. There are two statistical results.

1) As shown in Table III, NCD-CEA elapses the least time for solving Hard-EAP happening in all the three networks, and wins over other algorithms in two of three networks for solving Easy-EAP. In Table IV, as the network scale increases from 500 to 1000, the advantages of NCD-CEA in reducing the time complexity become more and more clear.
2) In small networks, the time costs of NCD-CEA for Easy-EAP and Hard-EAP are close to each other, but as the network size increases, the elapsed time of Hard-EAP is increased exponentially, while the elapsed time of Easy-EAP is increased linearly.

The solution space of both Easy-EAP and Hard-EAP is $O(2^N)$, with N representing the network size. As shown in Fig. 4, different from Easy-EAP whose objective can be calculated in polynomial time, the solving time complexity of the objectives of Hard-EAPs is $O(N^2)$. It illustrates that the hard objective of Hard-EAP is still a bottleneck for the application of algorithms on the super-large-scale networks with tens of thousands of nodes. Solving such a bottleneck needs further investigations in the effective methods for computing maximum eigenvalues.

To conclude, compared to most EAs, NCD-CEA can provide higher quality solutions using less computational cost. Those advantages are especially significant for the problems happening in large-scale networks.

D. Effectiveness in Virus Control

The final goal of this article is to prevent virus spread. Existing studies [79] have provided both the theoretical and experimental evidences that as long as the algorithm takes priority in comparison experiments, it would take advantage of epidemic control. Since NCD-CEA performs better than the other algorithms in comparison experiments, it should perform best in actual virus control theoretically.

To verify this assumption, we investigate the virus spread dynamically after resource intervention in three practical networks (Ps-contact, Ex-contact, and Email). Three algorithms with good performance are compared: 1) CLPSO; 2) MVBPSO; and 3) NCD-CEA, taking λ as the solution evaluation criterion. Besides, we also consider a random solution (called as "Random") and an empty solution (called as "None"). A population of feasible solutions is generated through the initialization process of NCD-CEA, and then the best one in them is selected as the random solution. The none solution represents the situation of no resource intervention. As the solutions determine the resource allocation, the variable of SEIV model in the network will be changed. Then, we evolve the SEIV model for 300 times to observe the virus spread dynamically. The experimental results are obtained by averaging over 20 independent runs. As is shown in Fig. 5, the number of infectious nodes is at a high level if without any resource intervention. The resource distribution, even with a random solution, can significantly decrease the number of infectious nodes. The higher quality is the solution, the better is the virus prevention effect. According to Tables III and IV, for Hard-EAP, NCD-CEA performs best in minimizing the optimization objective. Accordingly, the algorithm which leads to the fewest infectious individuals is NCD-CEA, followed by MVBPSO and CLPSO.

It can be observed from the experimental results that a better solution with a smaller λ can usually achieve a better control effect. Since NCD-CEA outperforms the other algorithms used in terms of minimizing λ, it also has the best performance in virus control.

VI. Conclusion

In this article, we consider the discrete resources to imitate the real-world resources. The virus spreading control, therefore, becomes typical subset selection optimization problems which are nondifferentiable, nonconvex, and NP-hard. To solve the problems, we propose an evolutionary divide-and-conquer algorithm (namely, NCD-CEA), in which a network-community-based decomposition strategy and an alternative

evolution process are designed to improve its efficiency and produce high-quality solutions.

The major topic of this article is dividing and conquering the virus spreading control problems, so we only focus on the theoretical framework of the algorithm. There are still many aspects waited to undergo. First, as a general algorithm framework, NCD-CEA utilizes community detection techniques to divide the problem and the solution space. But if the network is highly coupled, namely, the community structure is not significant, NCD-CEA will devolve into its basic optimizer inside and lose the superiority. Second, the default optimizer of NCD-CEA is originally designed for discrete optimization. Facing continuous optimization, who is the most suitable optimizer is not tested and discussed in this article.

In the future, we hope to explore a more collaborative way in coordinating the evolution of subswarms, to adapt the highly coupled networks. Thereafter, the basic continuous optimizers for NCD-CEA will be explored to widen its application fields.

REFERENCES

[1] J. A. Torres, S. Roy, and Y. Wan, "Sparse resource allocation for linear network spread dynamics," *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 1714–1728, Apr. 2017.

[2] S. Bhatt *et al.*, "The effect of malaria control on Plasmodium Falciparum in Africa between 2000 and 2015," *Nature*, vol. 526, no. 7572, pp. 207–211, 2015.

[3] I. Tomovski and L. Kocarev, "Simple algorithm for virus spreading control on complex networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 4, pp. 763–771, Apr. 2012.

[4] Z. Chen and C. Ji, "Spatial–temporal modeling of malware propagation in networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 5, pp. 1291–1303, Sep. 2005.

[5] C. M. Schneider, T. Mihaljev, S. Havlin, and H. J. Herrmann, "Suppressing epidemics with a limited amount of immunization units," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 84, no. 6, 2011, Art. no. 061911.

[6] R. Cohen and S. Havlin, "Efficient immunization strategies for computer networks and populations," *Phys. Rev. Lett.*, vol. 91, no. 24, 2013, Art. no. 247901.

[7] K. Drakopoulos, A. Ozdaglar, and J. Tsitsiklis, "An efficient curing policy for epidemics on graphs," *IEEE Trans. Netw. Sci. Eng.*, vol. 2, no. 1, pp. 67–75, Jul.–Dec. 2014.

[8] H. Zheng and J. Wu, "Effective network quarantine with minimal restrictions on communication activities," *IEEE Trans. Netw. Sci. Eng.*, vol. 3, no. 3, pp. 159–170, Jul.–Sep. 2016.

[9] Q. Zhang, L. Zhong, S. Gao, and X. Li, "Optimizing HIV interventions for multiplex social networks via partition-based random search," *IEEE Trans. Cybern.*, vol. 48, no. 12, pp. 3411–3419, Dec. 2018.

[10] W.-N. Chen, D.-Z. Tan, Q. Yang, T. Gu, and J. Zhang, "Ant colony optimization for the control of pollutant spreading on social networks," *IEEE Trans. Cybern.*, early access, doi: 10.1109/TCYB.2019.2922266.

[11] X.-J. Li, C. Li, and X. Li, "Minimizing social cost of vaccinating network SIS epidemics," *IEEE Trans. Netw. Sci. Eng.*, vol. 5, no. 4, pp. 326–335, Dec. 2018.

[12] E. Ramirez-Llanos and S. Martinez, "Distributed and robust fair optimization applied to virus diffusion control," *IEEE Trans. Netw. Sci. Eng.*, vol. 4, no. 1, pp. 41–54, Mar. 2017.

[13] G. Theodorakopoulos, J.-Y. Le Boudec, and J. S. Baras, "Selfish response to epidemic propagation," *IEEE Trans. Autom. Control*, vol. 58, no. 2, pp. 363–376, Feb. 2013.

[14] P.-Y. Chen, S.-M. Cheng, and K.-C. Chen, "Optimal control of epidemic information dissemination over networks," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2316–2328, Dec. 2014.

[15] C. Borgs, J. Chayes, A. Ganesh, and A. Saberi, "How to distribute antidote to control epidemics," *Random Struct. Algorithms*, vol. 37, no. 2, pp. 204–222, 2010.

[16] K. Scaman, A. Kalogeratos, and N. Vayatis, "Suppressing epidemics in networks using priority planning," *IEEE Trans. Netw. Sci. Eng.*, vol. 3, no. 4, pp. 271–285, Oct.–Dec. 2016.

[17] R. Bronfman-Nadas, N. Zincir-Heywood, and J. T. Jacobs, "An artificial arms race: Could it improve mobile malware detectors?" in *Proc. IEEE Netw. Traffic Meas. Anal. (TMA) Conf.*, 2018, pp. 1–8.

[18] H. G. Kayacık, A. N. Zincir-heywood, and M. I. Heywood, "Can a good offense be a good defense? Vulnerability testing of anomaly detectors through an artificial arms race," *Appl. Soft Comput.*, vol. 11, no. 7, pp. 4366–4383, 2011.

[19] S. Sen, E. Aydogan, and A. I. Aysan, "Coevolution of mobile malware and anti-malware," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2563–2574, Oct. 2018.

[20] G. Meng *et al.*, "Mystique: Evolving Android malware for auditing anti-malware tools," in *Proc. ACM Asia Conf. Comput. Commun. Security*, 2016, pp. 365–376.

[21] Y. Xue *et al.*, "Auditing anti-malware tools by evolving Android malware and dynamic loading technique," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 7, pp. 1529–1544, Jul. 2017.

[22] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Optimal resource allocation for control of networked epidemic models," *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 2, pp. 159–169, Jun. 2017.

[23] A. Beck, A. Nedić, A. Ozdaglar, and M. Teboulle, "An $O(1/k)$ gradient method for network resource allocation problems," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 1, pp. 64–73, Mar. 2014.

[24] S. Han, V. M. Preciado, C. Nowzari, and G. J. Pappas, "Data-driven network resource allocation for controlling spreading processes," *IEEE Trans. Netw. Sci. Eng.*, vol. 2, no. 4, pp. 127–138, Oct.–Dec. 2016.

[25] V. M. Preciado, M. Zargham, C. Enyioha, A. Jadbabaie, and G. J. Pappas, "Optimal resource allocation for network protection against spreading processes," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 1, pp. 99–108, Mar. 2014.

[26] Z. Deng, S. Liang, and Y. Hong, "Distributed continuous-time algorithms for resource allocation problems over weight-balanced digraphs," *IEEE Trans. Cybern.*, vol. 48, no. 11, pp. 3116–3125, Nov. 2017.

[27] A. Lobstein, "The hardness of solving subset sum with preprocessing," *IEEE Trans. Inf. Theory*, vol. 36, no. 4, pp. 943–946, Jul. 1990.

[28] W.-N. Chen, J. Zhang, H. S. H. Chung, W.-L. Zhong, W.-G. Wu, and Y.-H. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 278–300, Apr. 2010.

[29] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1344–1357, May 2016.

[30] Y. Zhang, X. Li, and A. V. Vasilakos, "Spectral analysis of epidemic thresholds of temporal networks," *IEEE Trans. Cybern.*, early access.

[31] A. Song, W.-N. Chen, T. Gu, H. Zhang, and J. Zhang, "A constructive particle swarm optimizer for virtual network embedding," *IEEE Trans. Netw. Sci. Eng.*, early access, doi: 10.1109/TNSE.2019.2932781.

[32] Q. Yang, W.-N. Chen, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 578–594, Aug. 2018.

[33] Q. Yang *et al.*, "Segment-based predominant learning swarm optimizer for large-scale optimization," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2896–2910, Sep. 2017.

[34] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech. Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.

[35] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. Int. Conf. Parallel Probl. Nat.*, 1994, pp. 249–257.

[36] X. Ma *et al.*, "A survey on cooperative co-evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 421–441, Jul. 2019.

[37] F. Van Den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[38] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[39] W.-N. Chen, Y.-H. Jia, F. Zhao, X.-N. Luo, X.-D. Jia, and J. Zhang, "A cooperative co-evolutionary approach to large-scale multisource water distribution network optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 842–857, Oct. 2019.

[40] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.

[41] L. Yang, X. Cao, D. Jin, X. Wang, and D. Meng, "A unified semi-supervised community detection framework using latent space graph regularization," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2585–2598, Nov. 2015.

[42] L. Yang, X. Cao, D. He, C. Wang, W. Xiao, and W. Zhang, "Modularity based community detection with deep learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 2252–2258.
[43] T. N. Dinh and M. T. Thai, "Community detection in scale-free networks: Approximation algorithms for maximizing modularity," *IEEE J. Sel. Areas Commu.*, vol. 6, no. 31, pp. 997–1006, Jun. 2013.
[44] M. E. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.
[45] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.
[46] M. E. Newman and E. A. Leicht, "Mixture models and exploratory analysis in networks," *Proc. Nat. Acad. Sci. USA*, vol. 104, no. 23, pp. 9564–9569, 2007.
[47] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, no. 2, 2004, Art. no. 26113.
[48] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New J. Phys.*, vol. 11, no. 3, 2009, Art. no. 33015.
[49] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 76, no. 3, 2007, Art. no. 036106.
[50] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger, "SCAN: A structural clustering algorithm for networks," in *Proc.13th ACM SIGKDD*, 2007, pp. 824–833.
[51] J. Wang, L. Wang, and X. Li, "Identifying spatial invasion of pandemics on arrival history," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 2782–2795, Dec. 2016.
[52] W. Liu, Z.-H. Deng, L. Cao, X. Xu, H. Liu, and X. Gong, "Mining top K spread sources for a specific topic and a given node," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2472–2483, Nov. 2015.
[53] S. Xu, P. Wang, C.-X. Zhang, and J. J. Lü, "Spectral learning algorithm reveals propagation capability of complex networks," *IEEE Trans. Cybern.*, vol. 49, no. 12, pp. 4253–4261, Dec. 2019.
[54] C. Sun and W. Yang, "Global results for an SIRS model with vaccination and isolation," *Nonlinear Anal. Real World Appl.*, vol. 11, no. 5, pp. 4223–4237, 2010.
[55] S. Trajanovski, Y. Hayel, E. Altman, H. Wang, and P. Van Mieghem, "Decentralized protection strategies against SIS epidemics in networks," *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 4, pp. 406–419, Dec. 2015.
[56] N. G. Becker, K. Glass, Z. Li, and G. K. Aldis, "Controlling emerging infectious diseases like SARS," *Math. Biosci.*, vol. 193, no. 2, pp. 205–221, 2005.
[57] T. Liu, X. Guan, Q. Zheng, and Y. Qu, "A new worm exploiting IPv6 and IPv4-IPv6 dual-stack networks: Experiment, modeling, simulation, and defense," *IEEE Netw. Mag. Global Internetw.*, vol. 23, no. 5, pp. 22–29, Sep. 2009.
[58] C. Gao, J. Liu, and N. Zhong, "Network immunization with distributed autonomy-oriented entities," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1222–1229, Jul. 2011.
[59] C. Griffin and R. Brooks, "A note on the spread of worms in scale-free networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 198–202, Feb. 2006.
[60] D. Trpevski, W. K. Tang, and L. Kocarev, "Model for rumor spreading over networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 81, no. 2, 2010, Art. no. 56102.
[61] M. Clerc. (2010). *Simplified Particle Swarm Optimiser for Binary Problems*. [Online]. Available: http://clerc.maurice.free.fr/pso/binary_pso/
[62] R. Pastorsatorras and A. Vespignani, "Epidemic spreading in scale-free networks," *Phys. Rev. Lett.*, vol. 86, no. 14, p. 3200, 2001.
[63] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
[64] V. Gemmetto, A. Barrat, and C. Cattuto, "Mitigation of infectious disease at school: Targeted class closure vs school closure," *BMC Infect. Disorders*, vol. 14, no. 1, pp. 695–705, 2014.
[65] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck, "What's in a crowd? Analysis of face-to-face behavioral networks," *J. Theor. Biol.*, vol. 271, no. 1, pp. 166–180, 2011.
[66] B. Zhang, R. Liu, D. Massey, and L. Zhang, "Collecting the Internet AS-level topology," *ACM SIGCOMM Comput. Commun.*, vol. 35, no. 1, pp. 53–61, 2005.
[67] X. Zeng, W. Chen, and J. Zhang, "An analysis of binary particle swarm optimizers for task assigning problem in wireless sensor networks," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2015, pp. 1974–1979.
[68] K. Kato and M. Sakawa, "Genetic algorithms with decomposition procedures for multidimensional 0–1 knapsack problems with block angular structures," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 33, no. 3, pp. 410–419, Jun. 2003.
[69] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A Stat. Mech. Appl.*, vol. 391, no. 6, pp. 2193–2196, 2012.
[70] Y. Liu *et al.*, "Solving NP-hard problems with physarum-based ant colony system," *IEEE/ACM Trans. Comput. Bio. Bioinfo.*, vol. 14, no. 1, pp. 108–120, Jan./Feb. 2017.
[71] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE CEC*, 2008, pp. 1110–1116.
[72] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.
[73] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
[74] P. E. McKight and J. Najab, "Kruskal–Wallis test," in *Corsini Encyclopedia of Psychology*. Hoboken, NJ, USA: Wiley, 2010, p. 1.
[75] W. Haynes, "Wilcoxon rank sum test," in *Encyclopedia of Systems Biology*. New York, NY, USA: Springer, 2013, pp. 2354–2355.
[76] M. Aickin and H. Gensler, "Adjusting for multiple testing when reporting research results: The Bonferroni vs. Holm methods," *Amer. Public Health Assoc.*, vol. 86, no. 5, pp. 726–728, 1996.
[77] M. Dorigo and L. M. Gambardella, "Ant colony system? A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[78] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2004.
[79] T.-F. Zhao, W.-N. Chen, A. W.-C. Liew, T. Gu, X.-K. Wu, and J. Zhang, "A binary particle swarm optimizer with priority planning and hierarchical learning for networked epidemic control," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, doi: 10.1109/TSMC.2019.2945055.
[80] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. IEEE Int. Conf. Comput. Cyber. Simulat.*, 1997, pp. 4104–4108.

Tian-Fang Zhao (Student Member, IEEE) received the master's degree from the Dalian University of Technology, Dalian, China, in 2017. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China.

Her current research interests include evolutionary computation, network propagation dynamics, and social network data analytics.

Wei-Neng Chen (Senior Member, IEEE) received the bachelor's and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

Since 2016, he has been a Full Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. He has coauthored over 100 international journal and conference papers, including more than 40 papers published in the IEEE TRANSACTIONS journals. His current research interests include computational intelligence, swarm intelligence, and network science and their applications.

Prof. Chen was a recipient of the IEEE Computational Intelligence Society (CIS) Outstanding Dissertation Award in 2016, and the National Science Fund for Excellent Young Scholars in 2016. He is currently the Vice-Chair of the IEEE Guangzhou Section. He is also a Committee Member of the IEEE CIS Emerging Topics Task Force. He serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and *Complex & Intelligent Systems*.

Sam Kwong (Fellow, IEEE) received the B.S. degree in electrical engineering from the State University of New York at Buffalo, Buffalo, NY, USA, in 1983, the M.S. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree from the University of Hagen, Hagen, Germany, in 1996.

From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada, Mississauga, ON, Canada. He joined Bell-Northern Research, Ottawa, ON, Canada, as a member of Scientific Staff. In 1990, he became a Lecturer with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, where he is currently a Professor with the Department of Computer Science. His research interests include video and image coding and evolutionary algorithms.

Tian-Long Gu received the M.Eng. degree from Xidian University, Xi'an, China, in 1987, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1996.

From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Perth, WA, Australia, and a Postdoctoral Fellow with the School of Engineering, Murdoch University, Perth. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, China. His research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Hua-Qiang Yuan received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 1996.

He is currently a Professor with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan, China. His current research interests include computational intelligence and cyberspace security.

Jie Zhang (Member, IEEE) received the master's degree in computer science from the China University of Mining and Technology, Xuzhou, China, in 1999.

She is currently an Associate Professor with the Beijing University of Chemical Technology, Beijing, China. Her current research interests include formal verification and PHM for power and embedded system design.

Ms. Zhang is a member of the Chinese Institute of Electronics Embedded Expert Committee.

Jun Zhang (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Professor of the Division Electrical Engineering, Hanyang University, Seoul, South Korea. His research interests include computational intelligence, cloud computing, data mining, and power electronic circuits. He has published over 200 technical papers in his research areas.

Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.