Higher Order Knowledge Transfer for Dynamic Community Detection With Great Changes
Huixin Ma, Kai Wu®, Member, IEEE, Handing Wang®, Member, IEEE, and Jing Liu®, Senior Member, IEEE

Abstract—Network structure evolves with time in the real world, and the discovery of changing communities in dynamic networks is an important research topic that poses challenging tasks. Most existing methods assume that no significant change occurs; namely, the difference between adjacent snapshots is slight. However, great change exists in the real world usually. The great change in the network will result in the community detection algorithms are difficulty obtaining valuable information from the previous snapshot, leading to negative transfer for the next time steps. This article focuses on dynamic community detection with substantial changes by integrating higher order knowledge from the previous snapshots to aid the subsequent snapshots. Moreover, to improve search efficiency, a higher order knowledge transfer strategy is designed to determine first-order and higher order knowledge by detecting the similarity of the adjacency matrix of snapshots. In this way, our proposal can keep the advantages of previous community detection results and transfer them to the next task. We conduct the experiments on four real-world networks, including the networks with great or minor changes. Experimental results in the low-similarity datasets demonstrate that higher order knowledge is more valuable than first-order knowledge when the network changes significantly and keeps the advantage even if handling the high-similarity datasets. Our proposal can also guide other dynamic optimization problems with great changes.

Index Terms—Dynamic community detection (DCD), evolutionary algorithm (EA), multiobjective optimization, transfer learning.

I. INTRODUCTION

COMMUNITY detection plays a crucial role in representing many real-world complex systems [1], [2], [3]. In finding out the change of networks over time, researchers introduce a concept, dynamic community detection (DCD).

Manuscript received 13 July 2022; revised 16 October 2022 and 20 January 2023; accepted 7 March 2023. Date of publication 15 March 2023; in part by the National Natural Science Foundation of China under Grant 62206205 and Grant 61976165; in part by the Fundamental Research Funds for the Central Universities under Grant JJS211905; in part by the Zhejiang Lab’s International Talent Fund for Young Professionals; in part by the Guangdong High-Level Innovation Research Institution Project under Grant 2021B090605008; and in part by the Guangzhou Key Research and Development Program under Grant 202206030003. (Corresponding author: Kai Wu.)

Huixin Ma, Kai Wu, and Handing Wang are with the School of Artificial Intelligence, Xidian University, Xi’an 710071, China (e-mail: 948181384@qq.com; kwu@xidian.edu.cn; hdwang@xidian.edu.cn).
Jing Liu is with the Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China (e-mail: neouma@mail.xidian.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TEVC.2023.3257563.
Digital Object Identifier 10.1109/TEVC.2023.3257563

Dynamic communities capture the network structures varying with time. Dynamic interaction graphs are applied ubiquity in many fields, such as social networks [4] and protein networks [5].

For this reason, there are many studies aimed at finding and studying this structure, such as spectral clustering [6], [7], deep learning [8], [37], [45], and non-negative matrix factorization [18]. In general, the DCD problem is NP-hard, and then many efforts have employed evolutionary algorithms (EAs) to solve this problem effectively, and EAs have become one of the leading solutions [10], [11]. How to promote the performance of EAs on this problem is also the main focus of this article. Aynaud and Gaillourme [9] proposed a static community detection algorithm for evolving networks. A pattern of community detection with temporal smoothness is formulated as a multiobjective problem [10], [11]. A diffusion model is presented in [12] for evolving networks. Evolutionary-based clustering approaches are introduced to maximize cluster accuracy and minimize the temporal cost from one time step to the successive one [10], [13], [14], [15]. However, in the real world, the communities may keep their topological features altered dramatically in the process of evolution, and some types of networks with changes after perturbing are introduced [20]. Great changes occur when nodes and edges are removed, added, and changed in the network beyond a certain threshold. Although existing algorithms have demonstrated superior performance in dynamic networks with minor changes, there is no guarantee that those algorithms still have reasonably good performance in dynamic networks with great changes.

To show the impact of the great network changes on the performance of algorithms, we run the representative DYNMOGA [10] algorithm at the 1st, 5th, and 10th time steps in the SYN-FIX dataset. As shown in Fig. 1, DYNMOGA has greater accuracy in the successive time steps than in the intermittent time step. It reveals that DYNMOGA may neglect the negative effect of the great changes, and it is of great significance that pays attention to this crucial point. It is believed that vertexes and edges do not change randomly in those snapshots. Generally, there is a hypothesis that the deviation from one time step to a successive one is negligible in most state-of-the-art algorithms. The performance of those algorithms decreases when the fact goes against the assumption.

These two characteristics inspire our solution: 1) on the one hand, it is feasible to utilize those unchanged features to improve the performance of the DCD problem and 2) on the other hand, the existing DCD algorithms neglect the network
with great changes, which leads to a negative transfer from the previous snapshot to the following snapshot. However, current methods only consider the first characteristic and ignore the second one, as shown in Fig. 1. Therefore, there is an urgent need to design a strategy to balance the impact of negative transfer and positive transfer on the DCD algorithms.

This article proposes a higher order knowledge transfer for a multiobjective genetic algorithm named HoKT (higher order knowledge transfer), integrating the designed higher order normalized mutual information (HoNMI) as one of the objective functions to overcome the shortcoming that the great changes can result in a negative transfer. Many DCD algorithms [10], [21], [22], [23] introduce a concept of knowledge to minimize the differences between the community structure at the current time step and that obtained at the previous time step. However, after the severe disturbance of the network, the knowledge hurts the performance of the algorithms. HoKT mainly overcomes this adverse effect by introducing the concept of higher order knowledge. We measure the overlapping degree between the current and previous snapshots when the network changes. This operation can reduce not only the weight of the first-order knowledge but also utilize the higher order knowledge.

First, we conduct experiments to confirm that low similarity exists. Second, the experiments on synthetic and real-life networks show that HoKT performs better, including the convergence rate, accuracy, and efficiency than the existing algorithms in solving the dynamic community problem with minor changes. Moreover, we transform four datasets, SYNFIX, Enron, Cell Phone Call, and Four Events, as networks with great changes. The experimental results show that HoKT has an excellent performance in the network with great changes.

The main contributions of HoKT are summarized as follows.

1) Great changes often occur in dynamic networks. A large amount of negative transfer degrades the current community detection algorithms’ performance because they assume that adjacent snapshots have high similarity. This article is the first work to focus on the DCD problems with great changes and then give a simple but effective solution.
2) We propose the higher order knowledge transfer strategy to reduce the negative effect but enhance the positive effect when great changes occur by measuring the overlapping degree between two adjacent snapshots of communities.
3) The proposed strategy also has a good reference for other dynamic optimization problems with great changes. We could consider how to utilize the results of early timestamps to achieve higher order knowledge transfer so that it balances the effect of negative and positive transfer.

The remainder of this article is organized as follows. The backgrounds of multiobjective optimization, and the review of DCD methods and transfer learning for dynamic multiobjective optimization problems (DMOPs) are presented in Section II. Section III describes the proposed approach, the HoKT, including the objective functions and operators used. Experimental results are reported in Section IV. Section V concludes our proposal and discusses future work.

II. RELATED WORK

Our proposal mainly focuses on transferring helpful knowledge of the previous tasks to promote the performance of the following tasks. This section introduces the current DCD methods. Most of them can be classified as exploiting transfer learning to solve the problem of dynamic multiobjective community detection. Moreover, we introduce the basic definitions of multiobjective optimization. It is inefficient to detect the current dynamic community using previous knowledge when there is a low similarity between the current and previous networks. This article aims to reduce the negative transfer effect when great changes occur. We also show the studies of transfer learning for DMOPs.

A. Multiobjective Optimization

Since most of the current evolutionary-based DCD algorithms model the DCD problem as a multiobjective optimization problem (MOP), our proposal is also multiobjective optimization. This section introduces the basic knowledge of MOPs. Evolutionary multiobjective optimization, whose primary goal is to deal with MOPs, has attracted increasing interest in the evolutionary computation community [48], for example, the nondominated sorting-based multiobjective evolutionary algorithm (NSGA) [49].

Generally, a MOP with $n$ decision variables and $m$ objective variables can be formulated as follows:

$$
\begin{align*}
\min & \quad y = F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T \\
\text{s.t.} & \quad g_i(x) \geq 0, i = 1, 2, \ldots, q \\
& \quad h_j(x) = 0, j = 1, 2, \ldots, p
\end{align*}
$$

(1)

where $x = (x_1, x_2, \ldots, x_n) \subseteq X \in \mathbb{R}^n$ is a vector of design variables, and $n$ is the number of independent variables $x_i$. $y = (y_1, y_2, \ldots, y_m) \subseteq Y \in \mathbb{R}^m$ is a vector of objective functions. $Y$ is the target space. $F(x)$ maps the solution space into
which further improved the efficiency and effectiveness of the propagation method to initialize the communities and restrict DCD (DYNMODPSO). To consider the nonlinear properties on low-dimensional representation. Otherwise, two studies to tackle the problem of discovering dynamic communities in weighted graphs are proposed in [46] and [54]. Huang et al. [47] investigated a compact and elegant community pattern based on the k-truss concept, which has advantages in computational costs and efficiency over previous algorithms to update the communities in both large and dynamic networks.

Although significant efforts have been made to discover dynamic communities, there are still drawbacks. The algorithms assume that the abrupt change in the network does not occur. The above algorithms either do not integrate the knowledge into the DCD problem or consider that the network structures of two adjacent time steps always tend to be similar in the transfer learning process. None of them consider the network changes’ negative effect. In order to eliminate the negative impact, our algorithm automatically selects high-order knowledge or low-order knowledge to detect dynamic communities. First-order knowledge plays an important role when the network structure does not change abruptly; in contrast, high-order knowledge reduces the negative effect of knowledge when the network changes significantly.

B. Dynamic Community Detection Methods

Individuals in the networks interact in diverse ways. A central issue in studying societies is identifying communities [2]. Membership in networks frequently changes, thus significantly characterizing a community structure in dynamic networks. Several ways have been proposed to detect communities in dynamic networks [38], [39], [40], [41]. Several methods for detecting communities in dynamic networks offered before 2010 were described in [42]. Convenient methods for detecting communities in dynamic networks are to view the network as snapshots with small changes based on the time stamps and employ static algorithms directly on each snapshot [38], [43]. However, using static algorithms repeatedly is computationally expensive, especially, when the number of snapshots is large. Accordingly, another way to detect the communities is using the feature of the network varieties.

Inspired by the evolutionary clustering framework, Folino and Pizzuti [10] proposed dynamic multiobjective genetic algorithms (DYNMOGAs) to detect the community structure in dynamic networks. The first objective is to maximize the modular structures at the current time step, and the second objective aims to minimize the divergence between the community structure at the current time step and that obtained at the previous time step. Zeng et al. [16] introduced the concept of consensus community to avoid the preceding issue of DYNMOGA. In this way, they guide the current population toward a direction similar to the community structure at the previous time step. Niu et al. [17] employed the label propagation method to initialize the communities and restrict the conditions of the mutation process of genetic algorithms, which further improved the efficiency and effectiveness of community detection. Liu et al. [21] discovered some drawbacks with respect to DCD. Accordingly, they developed a migration operator cooperating with the classic genetic operators to search for intercommunity connections. To overcome the drawbacks of community detection, such as the absence of error correction and the NP-hard of modularity-based community detection, Yin et al. [44] proposed an effective multiobjective particle swarm optimization to handle DCD (DYNMODPSO). To consider the nonlinear characteristics of networks, Wang et al. [45] proposed a semisupervised algorithm to overcome the effects of nonlinear properties on low-dimensional representation.

C. Transfer Learning for DMOPs

In recent years, many advancements have been made in solving DMOPs based on transfer learning [24]. Jiang et al. [25] proposed an algorithmic framework that exploits the transfer learning technique as a tool to solve DMOPs. The problem is that low-quality individuals exist during migration, so obtaining substantial improvements in conventional transfer learning is difficult. To overcome the difficulty of the negative transfer, an imbalanced transfer learning method KT-DMOEA [26], is proposed by adjusting the weights of the solutions and employing the knee points. Jiang et al. [27] implemented a novel memory-driven manifold transfer learning-based EA for DMOPs, which combines the technique of memory to remain the best individuals from the past with the feature of manifold transfer learning to predict the optimal individuals in the new time step in the process of the evolution. Liu et al. [28] employed the framework of transfer learning-assisted multiobjective evolutionary clustering with decomposition to handle high-dimensional datasets. Zhang and Wang [29] combined the centroid distance into NSGA-III with transfer learning for DMOPs, named TC_NSGAIII. Liu and Wang [30] have characterized a transfer learning algorithm that retains the prediction method considering sufficient historical information and modifying the solutions provided by the population prediction strategy to construct the initial population for optimization in the new environment. Zhang et al. [55] proposed a new relatedness metric based on the behavior distributions of the variable-length genetic programming individuals that choose the most suitable assisted tasks to transfer knowledge. Two novel operations (clustering-based transfer learning method and clustering-based transfer) are suggested in [31]
to reduce negative transfer and find more effective transferred solutions. Jiang et al. [32] proposed an individual transfer-based dynamic multiobjective EA, which filters out some high-quality individuals from historical optimal solutions to avoid the negative transfer. Some approaches aim to overcome the obstacle of low accuracy in the linear correlation instances and insufficient training samples. For example, Xu et al. [33] proposed an incremental support vector machine-based dynamic multiobjective EA. The DMOPs are treated as an online learning process, updating the support vector machine with the historical optimal solution. Considering the change of a DMOP that may occur in both decision and objective spaces, Zhou et al. [34] proposed an evolutionary search algorithm with multiview prediction that uses a kernelized autoencoding model in a reproducing Hilbert space to obtain the multiview prediction. Jiang et al. [32] proposed a prediction strategy by extending the autoencoding evolutionary search for solving the DMOP.

Most of the current transfer learning-based dynamic multiobjective EAs are applied to handle continuous optimization problems, such as [27], [35], and [36]. Few of them focus on solving the discrete optimization problem, i.e., they cannot be applied to detect dynamic communities. HoKT introduces the concept of higher order knowledge, using the previous knowledge indirectly. Moreover, our strategy can also guide other dynamic optimization problems with great changes.

III. HoKT

This section introduces the proposed HoKT. We first introduce the newly designed objective functions used in this article. Second, we describe the designed higher order knowledge transfer strategy and then present the essential parts of MOGA. Finally, the overall procedure of HoKT is elaborated.

A. Dynamic Community Detection With Great Changes

A static network is modeled as a graph \( G = (V, E) \), where \( V \) is a set of nodes and \( E \) is a set of edges, which connect two elements of \( V \). A dynamic network is a sequence \( G = \{G^1, G^2, \ldots, G^T\} \), where each \( G^t \) is a snapshot of nodes and connections among these nodes at time \( t \), and \( t = \{1, 2, \ldots, T\} \) is a finite set of time steps. The difference between \( G^t \) and \( G^{t-1} \) is that some nodes and edges are removed, or new nodes and edges could be added. The dynamic network changes are shown in Fig. 2. We introduce the following two cases: 1) slight network change and 2) great network change. Slight network change is a problem considered by most of the current methods. For example, in Fig. 2, there is only a slight change between time steps \( t-1 \) and \( t \). Great network change is a new scenario proposed in this article. The change between the two steps is dramatic. For example, in Fig. 2, the change between time steps \( t \) and \( t+? \) is that node three is deleted, but node eight is added. Furthermore, the edges (2, 3), (3, 4), and (2, 7) are removed, and the edge (2, 8) is appended. The community structure at time step \( t \) can be formulated as \( CR^t = \{CR^t_1, CR^t_2, \ldots, CR^t_k\} \), where \( CR^t \) is the community structure set at time step \( t \), each \( CR^t_k \) in \( CR^t \) is a portion of \( G^t \) composed of nodes in \( G^t \), and \( k \) is the number of communities. For each node \( V^t_i \in CR^t_k \in CR^t \), the term \( V^t_i \cap V^t_j = \emptyset \) is satisfied.

In general, in terms of overlapping ratio, if the difference between the adjacent snapshots is more significant than 0.01, we can call it a great change. At this time, the influence of negative transfer may become stronger. This is not a strict definition; we are just describing a phenomenon. Moreover, we find that the difference between the adjacent snapshots is less than 0.005 for current datasets (see Figs. 8–14).

B. New Objective Functions

The objective functions we consider are modularity [51] and the proposed HoNMI. Modularity is widely known as a criterion for evaluating the equality of community structure. A community with high modularity has a high link density among vertices in the same communities but a sparse link density among nodes from different communities. The modularity is defined as follows:

\[
Q = \sum_{s=1}^{k} \left[ \frac{l_s}{m} - \left( \frac{d_s}{2m} \right)^2 \right]
\]  

where \( l_s \) is the number of edges joining vertices inside the community \( CR^t_s \), \( d_s \) is the sum of the degrees of the nodes in \( CR^t_s \), and \( m \) is the number of edges in the network \( G^t \).

Indirect transfer learning is adopted in our algorithm. \( NMI(CR^t, CR^{t-1}) \) measures clustering drift from time step \( t \) to time step \( t-1 \). The NMI is frequently set as an objective in many studies to minimize the difference between different community structures. The hypothesis exists in those efforts: the communities should not sharply change between two successive time steps, and the evolution should be presented as a smooth transition. However, the existence of the hypothesis is a problem on its own due to the condition where clusters may change violently is not taken into account.

Thus, we design a strategy introducing HoNMI as an objective function, which can employ higher order knowledge (the community label not only from the last snapshot but also the previous snapshots) effectively to avoid negative transfer when the network abruptly changes. Since we found that the previous snapshots may have a high similarity with the current snapshot, HoNMI: 1) adopts the majority voting principle to remove the negative transfer effect of the last snapshot and 2) makes full use of the community label information of other...
Highly similar snapshots. HoNMI is shown as follows:

\[
\text{HoNMI} = \sum_i w(i) \cdot \text{NMI}(CR_i, CR^t), \quad i = 1, \ldots, t - 1
\]

where \(w(i)\) refers to the weight at time step \(i(i = 1, \ldots, t - 1)\) and \(\Sigma w(i) = 1\). The definition of NMI [10] is shown as follows:

\[
\text{NMI}(A, B) = \frac{-2 \sum_{i=1}^{c_A} \sum_{j=1}^{c_B} C_{ij} \log(C_{ij}/N/C_i C_j)}{\sum_{i=1}^{c_A} C_i \log(C_i/N) + \sum_{j=1}^{c_B} C_j \log(C_j/N)}
\]

where \(A = \{A_1, A_2, \ldots, A_{c_A}\}\) and \(B = \{B_1, B_2, \ldots, B_{c_B}\}\) are the community label for two snapshots and \(c_{A(\cdot B)}\) is the number of communities in \(A(B)\). \(C\) is the confusion matrix whose element \(C_{ij}\) is the number of nodes in the community \(A_i \subseteq A\) that also appears in the community \(B_j \subseteq B\). \(C_i\) denotes the sum of the figures of \(C\) in row \(i\), and column \(j\) is marked \(C_{ij}\). The higher the NMI(CR^t, CR^t-1), the lower the temporal cost is. On the contrary, if there is no overlapping between CR^t and CR^t, NMI(CR^t, CR^t-1) = 0. NMI and Q are conflicting objectives (see supplementary material, Section III).

C. Higher Order Knowledge Transfer

We express first-order knowledge and higher order knowledge with HoNMI. The definition of first-order knowledge and higher order knowledge is shown as follows.

First-Order Knowledge: It computes the community label at time step \(t\) using the community label obtained at time step \(t - 1\). The objective function HoNMI is equal to NMI(CR^t-1, CR^t).

Higher Order Knowledge: It computes the community labels at time step \(t\) using community labels at previous time steps. The objective function HoNMI is equal to \(\Sigma \text{NMI}(CR^t, CR^t) \cdot w(i), i = 1, \ldots, t - 1\).

The overlapping ratio of the adjacency matrix at \(t\) to \(t - 1\) is calculated using the adjacency matrices at two successive time steps. We calculate the overlapping ratio as follows: ratio = numSameEdge/\(n\), where numSameEdge is the number of the edges that belong to two networks at the successive time steps and \(n\) is the number of edges at time step \(t\). Fig. 3 shows an example of computing the overlapping ratio. The edge set at time step \(t - 1\) is \(\{(1, 2), (1, 3), (1, 4), (2, 1), (3, 1), (3, 4), (4, 1), (4, 3)\}\) and the edge set at time step \(t\) is \(\{(1, 2), (1, 3), (2, 1), (2, 4), (2, 5), (3, 1), (3, 4), (4, 2), (4, 3), (4, 5), (5, 2), (5, 4)\}\). There are 8 and 12 edges at time steps \(t - 1\) and \(t\), respectively. The common edges that appear simultaneously in both networks are \(\{(1, 2), (1, 3), (2, 1), (3, 1), (3, 4), (4, 2)\}\). Thus, numSameEdge = 6, \(n = 12\), and the overlapping ratio \(r\) is obtained as 0.5. Algorithm 1 describes the steps to get it.

The ratio \(r < \sigma\) demonstrates that the negative transfer may happen if we use the first-order knowledge. Thus, we select the higher order knowledge to improve performance, where the decision-makers define the value of \(\sigma\). In this way, we indirectly use the higher order knowledge to enhance the performance of community detection at time \(t\), and the failure of direct use of historical knowledge also guarantees the diversity of initializing population. The current algorithm takes advantage of only the result of the previous snapshot. However, the current network shares similarities with all the networks preceding the current time step. Therefore, all of the results of the previous networks are available.

When the ratio \(r\) is greater than the threshold \(\sigma\) but the overlapping ratio between \(G^t\) and \(G^t - 1\) is smaller than \(\sigma\), HoNMI = NMI(CR^t, CR^t-1) is set as the second objective to detect communities. The first-order knowledge from the previous time snapshot is transferred. Furthermore, when the ratio \(r\) is less than the threshold \(\sigma\), we compute the NMI between \(t\) and all the previous time steps before time step \(t\) and then add them up based on the weight \(w\) to act as HoNMI, where \(w\) is decided by the similarity matrix. The selection strategy of \(w\) is shown in Section IV in detail.

D. Encoding

There are two categories representing clusters: label-based representation and locus-based representation. In label-based representation, a population consists of \(N\) solutions \(X = \{X_1, \ldots, X_N\}\), each including \(n\) genes \(g_1, g_2, \ldots, g_n\), where \(n\) is the number of nodes. If \(k\) is the number of the communities, the value of \(g_i\) is in the range \(\{1, \ldots, k\}\) that identifies the community to which node \(i\) belongs. Each gene can assume allele values \(g_i\) in the range \(\{1, \ldots, n\}\) in locus-based representation.

The value of \(g_i = j\) is interpreted as a connection between node \(i\) and \(j\). As shown in Fig. 4, nodes 1, 2, and 8 belong to the

![Diagram](image-url)
same community, and nodes 3, 4, 5, 6, and 7 belong to another community. In the label-based representation, the label of each node is the serial number of the community to which the node belongs. Thus, the label of nodes 1, 2, and 7 is 1, and the label of nodes 3, 4, 5, 6, and 7 is 2. In the locus-based representation, the label of a node is a node number belonging to the same community as the node. As a result, the label of node 1 can be set to 8 or 2, and node 2 is selected randomly as the final label of node 1.

A sharp advantage of the locus-based representation is that the number $k$ of clusters can be automatically determined by the number of components contained in an individual and obtained by the decoding step [37]. Thus, locus-based adjacency encoding is employed in our algorithm.

**E. Genetic Operators**

**Initialization:** The label of node $i$ is initialized by randomly choosing a neighbor from a data set. Each couple $(i, g_i)$ represents a link belonging to one of the components of $G$.

**Crossover:** Two individuals are given to be used as parents, and a random binary mask is created. The gene belonging to parent one is selected when the mask is 0, and the gene belonging to parent two is selected when the mask is 1. Those genes are chosen to form the genes of the child. The crossover process is shown in Fig. 5, and mask(1) = 1. As a result, the gene of parent two is selected as the gene of offspring, offspring(1) = parent2(1) = 5. The second element of the mask is 0; thus, the gene of parent one is selected as the gene of offspring, offspring(2) = parent1(2) = 3. The red elements are genes determined by the mask.

**Mutation:** Changing the value of the $g_i$ with one of the neighbors of node $i$ randomly. An example is shown in Fig. 6.

The original set of labels with locus-based representation is \{2, 8, 4, 7, 6, 3, 4, 2\}. The neighbors of node 1 are \{2, 8\} (node 1 is excluded). Then a node from the neighbor set is selected randomly as the label of node 1. Thus, node 8 is selected to form the label of node one after mutation. After performing the mutation operator, nodes 1, 6, 5, 3, 7, 4, and 1 are selected to become the labels 2, 3, 4, 5, 6, 7, and 8, respectively.

**Selection:** The binary tournament selection operator is applied in HoKT. According to the nondominated sorting strategy in NSGA-II, we first assign a rank to each individual, sort them according to nondomination rank, and then combine the parents and offspring into a new pool and partition them into fronts. Finally, we select points on the lower front and apply the mutation and crossover operator to produce the next population.

**F. Overall Procedure**

HoKT provides a solution to determine automatically whether to use high-order knowledge or first-order knowledge by obtaining the overlapping ratio of a snapshot to the previous shot. HoKT employs the nondominated sorting genetic algorithm (NSGA-II) [19] to optimize the designed objectives. It finds the communities of the first snapshot by optimizing only the first objective and generates the final clustering of the other snapshots by evaluating two objectives. We introduce two mechanisms to transfer knowledge, which are decided by the overlapping ratio of the adjacency matrix at \(t\) to \(t-1\).

The flowchart of HoKT is shown in Fig. 7. Given a dynamic network $G = \{G^1, \ldots, G^T\}$, HoKT finds a partitioning of $G^1$ by running the genetic algorithm that optimizes only (2). We use the locus-based adjacency representation to initialize and detect the network. Since no knowledge can be used at \(t = 1\), HoKT uses the genetic algorithm to detect communities with the first objective (2). When \(t = 2\), there is only first-order knowledge. The population that is the most similar to the previous community structure is selected from the initialized population. Thus, this method extracts the feature of the previous community structure from the current network. NMI is the metric to measure how similar the community
structure \( CR^t \) is to the previous clustering \( CR^{t-1} \). We use NMI as the objective function to select the most suitable population. The higher order knowledge utilization is carried out when \( t > 2 \). The overlapping ratio \( r \) between the current network \( G^t \) and the previous network \( G^{t-1} \) is obtained. The fact that \( r \) is low means that using first-order knowledge causes negative transfer learning, so we get the combination of NMI(\( CR^t, CR^{t-1} \)) and NMI(\( CR^t, CR^{t-2} \)) as the objective function, where \( i = 1, \ldots, t - 1 \). By this method, we can transfer higher order features to the current detection. Moreover, when \( r \) is high enough, it is believed that the first-order knowledge has much helpful information. Thus, first-order knowledge is still used to transfer knowledge.

A random population with \( n = W|V| \) individuals is created for a given number of time steps. Then, for a fixed number of generations, it decodes the individuals to generate the partitioning at time step \( t \) and evaluates the objective values. The crossover and mutation operators are used to create a new population. Parents and offspring are combined, and the new pool is partitioned into fronts. The selection operator is employed to assign a rank to each individual according to Pareto dominance and sorts them. The individuals with the lower rank are selected, and the crossover and mutation operators are applied to create the new population. At the end of each time step, HoKT returns a set of solutions, each corresponding to a different tradeoff between (2) and (4). According to DYNMOGA [10], we choose the partitioning with the highest modularity value. Algorithm 2 describes the detailed process of HoKT.

### Algorithm 2 HoKT

**Input:**
- \( G = \{ G^1, \ldots, G^T \} \): the sequence of graphs;
- \( T \): time steps;
- \( w_i \): the weight for higher-order knowledge transfer at time steps \( t \);
- \( \sigma \): the threshold;

**Output:**
- \( C = \{ C_1, C_2, \ldots, C_T \} \)

1. Initialize the clustering \( CR^1 = \{ C^1_1, C^1_2, \ldots, C^1_k \} \) of the network \( N1 \) by optimizing Eq. (2);
2. for \( t = 2 \) to \( T \) do
   3. Compute the overlapping ratio \( r \) of \( t \) to \( t - 1 \);
   4. if \( (r > \sigma) \) then
      5. \( \text{HoNMI} = \text{NMI}(t, t - 1) \);
   6. else
      7. \( \text{HoNMI} = \text{NMI}(t, t - 1) \times w_t \);
   8. end if
   9. Create a random population of individuals with the number \( n = |V| \);
10. Decode each individual to generate the partitioning \( CR^t = \{ C^t_1, \ldots, C^t_k \} \);
11. Evaluate the two fitness values of the translated individuals \( (Q, \text{NMI}) \);
12. while (termination criteria are not satisfied) do
   13. Create a population of offspring by applying the crossover and mutation operators;
   14. Decode each individual to generate the partitioning \( CR^t = \{ C^t_1, \ldots, C^t_k \} \);
   15. Evaluate the two fitness values of the translated individuals \( (Q, \text{NMI}) \);
   16. Assign a rank to each individual and sort them according to non-domination rank;
   17. Combine the parents and offspring into a new pool and partition it into fronts;
   18. Select points on the lower front and apply the mutation and crossover operator on them to create the next population;
19. end while
20. Return the solution \( CR^t = \{ C^t_1, \ldots, C^t_k \} \) with the maximum modularity value;
21. end for

### IV. Experiment Results

#### A. Experimental Setup

1) **Performance Metric**: This article employs two metrics to evaluate the performance of community detection: NMI and \( F_1 \)-score. The NMI in the experimental results refers to the NMI(\( CR^t, CR^t_{\text{real}} \)), where \( CR^t_{\text{real}} \) is the real community label at time step \( t \). \( F_1 \)-score, a metric of classification problem, is the harmonic mean of precision and recall. \( F_1 \)-score is defined as follows:

\[
F_1 \text{-score} = 2 \times \frac{\text{TP}}{\text{TP} + \text{FN}} \times \frac{\text{TP}}{\text{TP} + \text{FP}}
\]

where TP is the number of labels predicted correctly, FN is the number of labels predicted to be negative but positive in the real case, and FP is the number of labels predicted positive but negative in the real case.

2) **Baseline Methods**: When considering the great network change scenario, the main goal of this article is to verify that the proposed higher order knowledge transfer strategy can effectively balance the effects of positive and negative
Most of the dynamic multiobjective community detection methods introduced in Section III-B are variants of DYNMOGA, and they do not consider how to eliminate the impact of great network changes. Therefore, we only choose DYNMOGA as the benchmark. At the same time, there are also nonevolutionary algorithms, as mentioned in [18], [41], and [45]. They do not consider the impact of great network changes, and they are also different from the optimizers of our scheme. Comparing with them cannot verify the motivation of our scheme, and it is a meaningless and unfair comparison. We can transfer the ideas of this article to other DCD methods or dynamic optimization problems; however, doing so deviates from our original intention, which can be regarded as future work.

Most dynamic multiobjective algorithms based on transfer learning introduced in Section III-C are solutions for continuous optimization problems, especially, their knowledge transfer strategies that are difficult to solve the problem of DCD directly. Meanwhile, these schemes rarely consider the impact of great changes. We find that a knee point-based transfer learning method called KT-MOEA/D [26] can handle the problem of DCD. Therefore, we selected KT-MOEA/D as a representative algorithm to verify the effectiveness of our scheme.

DYNMOGA and KT-MOEA/D [26] are employed as baselines. KT-MOEA/D and DYNMOGA optimize (2) and (3), using only first-order knowledge. Moreover, KT-MOEA/D employs transfer learning to initialize the population of the subsequent snapshots.

3) Parameters: Setting parameters is a significant challenge for whatever algorithms we use. HoKT has five parameters to control: population size, generations, crossover probability, mutation probability, the order of knowledge (\(w\)), and \(\sigma\). The parameters are shown in Table I. The crossover, mutation, and selection operators of KT-MOEA/D and DYNMOGA are the same as HoKT. For each dataset, 30 independent runs are performed for all methods, and the Wilcoxon rank-sum test is employed to test the significance of the results.

### Table I

Parameters of HoKT

| Parameters                        | Value |
|-----------------------------------|-------|
| Population size                   | 200   |
| Generations                       | 100   |
| Crossover Probability             | 0.8   |
| Mutation Probability              | 0.2   |
| The order of knowledge            | 1, 2, 3 |

B. Cell Phone Call

The first dataset we applied is a real-life dynamic network, the Cell Phone Call dataset. This dataset consists of cell phone call records for ten days in June 2006, in which each of the nodes represents cell phones, and an edge increases with the connection between the two cell phones. There are 400 nodes and five core points in this network. The comparison between DYNMOGA and our algorithm shows the advantage of higher order knowledge, as shown in Table II. The best NMI or \(F_1\)-score values are marked with bold black font. In addition, we summarize the significance test results with the NMI and \(F_1\)-score values at the bottom row. The similarity for the Cell Phone Call dataset is shown in Fig. 8. The element \((i, j)\) is the overlapping ratio between the \(i\)th and \(j\)th snapshots.

HoKT tends to have the best performance over snapshots 3–9. This phenomenon is because the longer the time, the higher order knowledge is available. HoKT mainly employs second-order knowledge and third-order knowledge. We set \(w = \{0.6, 0.4\}, \{0.7, 0.3\}, \{0.8, 0.2\}, \{0.9, 0.1\}\) to transfer the second-order knowledge and \(w = \{0.5, 0.3, 0.2\}, \{0.6, 0.3, 0.1\}, \{0.7, 0.2, 0.1\}\) to transfer the third-order knowledge. We select the best results for the final presentation.

No other knowledge is used at the first step, and all three algorithms detect communities with static methods. As a result, HoKT gets a poor grade. At time step 2, HoKT, DYNMOGA, and KT-MOEA/D use first-order knowledge. The NMI of HoKT is worse than KT-MOEA/D but better than DYNMOGA. At time step 3, the second-order knowledge is used, and \(w = \{0.8, 0.2\}\), HoKT shows its advantages and gets the best NMI and \(F_1\)-score. The overlapping ratio \(r_{13}\) of snapshots between \(t = 1\) and \(t = 3\) is 0.7506 and \(r_{23} = 0.7600\). The difference between those two \(r\) values is noticeable, and \(r_{23}\) is greater than \(r_{13}\). Thus, the weight for \(t = 2\) is set to 0.8, and the weight for \(t = 1\) is 0.2.

The overlapping ratio \(r_{34}\) of snapshots between \(t = 4\) and \(t = 3\) is 0.7555 and \(r_{24} = 0.7540\). The difference between them is small and only 0.0015. Thus, the second-order knowledge is used, and \(w = \{0.6, 0.4\}\) at time step 4. At time step 5, the network at \(t = 5\) is more similar to the network at \(t = 4\) than at \(t = 3, 2, 1\). The weight for the fourth snapshot is greater than the third snapshot. Thus, the second-order knowledge is only used, and \(w = \{0.8, 0.2\}\). At time step 6, the third-order knowledge is used and \(w = \{0.7, 0.2, 0.1\}\). At time step 7, \(r_{67}\) is 0.7635, \(r_{57}\) is 0.7551, and the overlapping ratios between the seventh snapshot and other snapshots are smaller than \(r_{57}\). Thus, we select the second-order knowledge, and \(w = \{0.9, 0.1\}\). Similarly, at time step 8, the second-order knowledge is used, and \(w = \{0.8, 0.2\}\) due to \(r_{78} = 0.7640\) and \(r_{68} = 0.7542\). The large overlapping ratio illustrates more helpful knowledge in this snapshot for the following snapshot. At time step 9, HoKT uses the
TABLE II
NMI AND $F_1$-SCORE FOR THE CELL PHONE CALL DATASET WITH MINOR CHANGES

| Time | NMI | $F_1$-score |
|------|-----|-------------|
|      | HoKT | DYNMOGA | KT-MOEA/D | HoKT | DYNMOGA | KT-MOEA/D |
| t = 1 | 0.442±0.06 | 0.349±0.12 | 0.486±0.06 | 0.414±0.09 | 0.378±0.08 | 0.431±0.08 |
| t = 2 | 0.744±0.00 | 0.734±0.04 | 0.756±0.05 | 0.738±0.00 | 0.706±0.06 | 0.705±0.07 |
| t = 3 | 0.758±0.00 | 0.700±0.03 | 0.754±0.04 | 0.745±0.00 | 0.712±0.04 | 0.701±0.08 |
| t = 4 | 0.545±0.00 | 0.539±0.05 | 0.520±0.05 | 0.545±0.00 | 0.538±0.07 | 0.475±0.11 |
| t = 5 | 0.654±0.08 | 0.636±0.07 | 0.635±0.07 | 0.631±0.00 | 0.656±0.09 | 0.578±0.05 |
| t = 6 | 0.397±0.00 | 0.710±0.05 | 0.736±0.05 | 0.733±0.07 | 0.692±0.06 | 0.726±0.04 |
| t = 7 | 0.654±0.00 | 0.622±0.06 | 0.622±0.09 | 0.651±0.00 | 0.587±0.11 | 0.570±0.09 |
| t = 8 | 0.713±0.06 | 0.717±0.05 | 0.645±0.08 | 0.710±0.07 | 0.696±0.05 | 0.636±0.10 |
| t = 9 | 0.618±0.00 | 0.589±0.07 | 0.602±0.06 | 0.614±0.00 | 0.517±0.09 | 0.598±0.07 |
| t = 10 | 0.625±0.00 | 0.606±0.03 | 0.638±0.04 | 0.607±0.00 | 0.588±0.06 | 0.598±0.04 |

TABLE III
NMI FOR CELL WITH GREAT CHANGES

| Time | NMI | $w$ |
|------|-----|-----|
|      | HoKT | DYNMOGA | KT-MOEA/D |
| t = 1 | 0.394±0.01 | 0.401±0.00 | {1} |
| t = 4 | 0.665±0.00 | 0.624±0.01 | {1} |
| t = 5 | 0.612±0.00 | 0.624±0.00 | {1} |
| t = 7 | 0.621±0.00 | 0.565±0.00 | {0.7, 0.2, 0.1} |
| t = 9 | 0.568±0.00 | 0.536±0.00 | {0.7, 0.2, 0.1} |

C. Enron

The second dataset is also a real-life network that collects email records of a company in the U.S. from 1999 to 2002, called the Enron email dataset, which provides real-world data that is arguably of the same kind as data from Echelon intercepts—a set of messages about a wide range of topics, from a large group of people who do not form a closed set [53]. The network we used consists of 50 000 messages among 151 users. The similarity for the Enron dataset is shown in Fig. 9.

Table IV shows the NMI and $F_1$-score values of HoKT, KT-MOEA/D, and DYNMOGA. HoKT has a better NMI value than other algorithms in steps 2–12. As can be observed from the significant test results, HoKT obtains a competitive average NMI over DYNMOGA and KT-MOEA/D on 11 timesteps. In time step 1, we still detect communities by the static method, which is the same as KT-MOEA/D and DYNMOGA. HoKT prefers to obtain a high $F_1$-score during steps 2–4 and 6–10.

To obtain results in the network with great changes, we run HoKT and DYNMOGA in the first, third, fifth, seventh, and ninth time steps of the Cell Phone Call dataset, which is displayed in Table III. HoKT achieves the best solutions compared with DYNMOGA. It is noted that HoKT selects poor results with the static method at the first step in order to demonstrate its effectiveness. However, HoKT uses third-order knowledge and $w = \{0.7, 0.2, 0.1\}$ when $t = 7$ and 9. It leads to a decrease in negative transfer learning of lower order knowledge. At time step 5, since the network structure changes slightly, DYNMOGA obtains the better NMI value. The problem needs to be improved, i.e., how to obtain good results with little time steps.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
changes. It indicates that DYNMOGA has a poor upper limit in the network with great changes.

### D. SYNFIX

This section compares HoKT with KT-MOEA/D and DYNMOGA on a dataset with an unchanged number of communities called SYNFIX. The network contains four communities, each of which holds 32 nodes. Every node connects with the other 16 nodes and uses a number $z$ of links in conjunction with nodes else. Three nodes are selected and moved to the other three communities at each time step. Fig. 10 shows the similarity for the SYNFIX dataset.

Table VI shows NMI and $F_1$-score for SYNFIX when $z = 5$.

When $t = 2$, the first-order knowledge is used. The superior performances of HoKT compared to DYNMOGA and KT-MOEA/D are on 8 time steps. The similarity rates of networks on this dataset increase over time. When $t = 9$ and 10, all of the third-order knowledge is useful based on the similarities of the networks, and the weight is close.

As shown in Table VI, HoKT performs better than the other algorithms in all eight snapshots regarding NMI. NMI of HoKT is the same as that of DYNMOGA and KT-MOEA/D at time steps 3, 4, and 10. Although KT-MOEA/D has good performance at time step 4, it needs extra memory to store the knee points due to its specific transfer learning. Moreover, all the solutions obtained by HoKT when $t = 2, 3, 4, 5, 7,$ and 10 are equal to the ground truth. It is shown that our algorithm can achieve the best and most stable results in the SYNFIX dataset.

The NMI values in SYNFIX dataset with great changes is shown in Table VII. As mentioned above, no available knowledge can be obtained when $t = 1$. At $t = 3$, both two algorithms use first-order knowledge. At time step 5, the second-order knowledge is used, and $w = \{0.9, 0.1\}$ because the overlapping ratio $r_{35}$ is greater than $r_{15}$. At time step 7, the second-order knowledge is also used for simplicity, and $w = \{0.7, 0.3\}$ due to the relatively high similarity between $r_{57}$ and $r_{37}$. We can obtain the same claim at time step 9. When $t = 3, 5, 7,$ and 9, significant evolution is detected, and HoKT utilizes higher order knowledge to obtain the maximum value of NMI. The performance of HoKT is improved by about 0.4% compared to DYNMOGA.

### E. Four Events

The fifth dataset consists of four kinds of main events. There are five time steps in each snapshot. The descriptions of these datasets are shown as follows.

1) **Birth and Death**: 10% of communities are restructured by selecting nodes from other existing communities and removing nodes from origin clusters.

2) **Expansion and Contraction**: 10% of communities are expanded or reduced by 25% of the original scale.

3) **Intermittent Communities**: 10% of communities are concealed and present in the next step.

4) **Merging and Splitting**: Parts of the different communities are selected to merge at each time step.

Four datasets are generated based on four kinds of events. Each network consists of 1000 nodes whose mean degree is 15. In order to compare HoKT with other algorithms in the network with great changes, we run algorithms only at time steps 1, 3, and 5. Figs. 11–14 show the similarity matrix for the Four Events dataset. Table VIII depicts the metric NMI on the four datasets using HoKT, DYNMOGA, and KT-MOEA/D. Table IX shows the values of the $F_1$-score for HoKT, DYNMOGA, and KT-MOEA/D.
When $t = 1$, HoKT uses the static community detection method by optimizing (2), and we select the worst NMI and $F_1$-score values to show the effectiveness of HoKT. Thus, it is reasonable that the performance of HoKT is worse than DYNAMOGA and KT-MOEA/D. When $t = 3$, the first-order knowledge is used. When $t = 5$, the first-order or the second-order knowledge is used, and $w$ is listed in the table. NMI and $F_1$-score at time step 5 are mainly focused because the higher order knowledge is used only in time step 5. Table VIII shows that HoKT outperforms or matches DYNAMOGA in 11 out of 12 cases, only losing once to DYNAMOGA at the first step. Furthermore, the $F_1$-score of HoKT is higher than those computed on the second and fourth networks. In the case of expansion and contraction communities, the NMI of DYNAMOGA and KT-MOEA/D only decreased by about 0.01 and 0.04, respectively, from time step 3 to time step 5. Since the similarities between neighboring snapshots are more significant than 0.95, there is much helpful knowledge on first-order knowledge. However, the NMI of HoKT remains increased over the third time step. As regards the fourth dataset, although DYNAMOGA and KT-MOEA/D obtained higher NMI values than HoKT in the first and second time steps, NMI values obtained by DYNAMOGA and KT-MOEA/D at time step 5 are lower than that obtained by HoKT. Moreover, the extra knee-based transfer strategy in KT-MOEA/D results in high performance in the Intermittent Communities dataset. For the other three datasets, HoKT is better than KT-MOEA/D. Thus, higher order knowledge transfer is effective.

![Fig. 11. Similarity for the Birth and Death dataset.](image1)

![Fig. 12. Similarity for the Expansion and Contraction dataset.](image2)

![Fig. 13. Similarity for the intermittent communities dataset.](image3)

![Fig. 14. Similarity for merging and splitting dataset.](image4)
TABLE IX
F₁-SCORE FOR FOUR DATASETS WITH GREAT CHANGES

| Time | HoKT | DYNMOGA | KT-MOEA/D | w |
|------|------|---------|-----------|---|
| t = 1 | 0.9520±0.0074 | 0.9615±0.0173 | 0.9550±0.0214 | {} |
| t = 3 | 0.9974±0.0030 | 0.9920±0.0077 | 0.9921±0.0102 | {} |
| t = 5 | 0.9959±0.0026 | 0.9964±0.0009 | 0.9933±0.0054 | (0.8, 0.2) |

2 t = 1 | 0.9555±0.0175 | 0.9520±0.0208 | 0.9714±0.0137 | {} |
| t = 3 | 0.9972±0.0042 | 0.9965±0.0043 | 0.9928±0.0064 | {} |
| t = 5 | 0.9960±0.0036 | 0.9811±0.0113 | 0.9927±0.0052 | (0.9, 0.1) |

3 t = 1 | 0.9494±0.0241 | 0.9550±0.0034 | 0.9492±0.0332 | {} |
| t = 3 | 0.8512±0.0018 | 0.8507±0.0003 | 0.8778±0.0077 | {} |
| t = 5 | 0.6560±0.0020 | 0.6547±0.0015 | 0.7226±0.1551 | (0.9, 0.1) |

4 t = 1 | 0.9480±0.0231 | 0.9637±0.0048 | 0.9588±0.0079 | {} |
| t = 3 | 0.9922±0.0042 | 0.9931±0.0075 | 0.9903±0.0028 | {} |
| t = 5 | 0.9807±0.0089 | 0.9754±0.0158 | 0.9789±0.0121 | (0.7, 0.3) |

Fig. 15. Change of NMI for different thresholds.

F. Parameters Analysis

1) Effect of σ on HoKT: In this article, we determine σ by operating trial-and-error experiments on datasets. We set σ = {0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9} to run the algorithm on the benchmark merging and splitting dataset. The results are shown in Fig. 15. The red curve in Fig. 8 has the best NMI at each time step. The red data occur when σ = 0.7, 0.8, and 0.9. That is because the similarities of the network are around [0.7, 0.9]. Furthermore, only when σ = 0.7, 0.8, or 0.9, different order knowledge can be used. Furthermore, there are two red data when σ = 0.8. Thus, we set σ = 0.8. Moreover, we mainly migrate the second- and third-order knowledge and give weight to the different knowledge. We choose the best solution with different weights and orders as the final solution of HoKT.

It is concluded that the selection of σ is based on the network similarity. First, the similarity is computed to determine the value interval of σ. And then, the value of σ is obtained by operating trial-and-error experiments. If more than two snapshots have a high overlapping ratio with the current snapshot, we can use higher order knowledge; otherwise, we use first-order knowledge. It should be noted that if the similarity of two adjacent snapshots is low, we should use zero-order knowledge, i.e., not use any previous results.

2) Effect of w on HoKT: We study the influence of order and different weights on HoKT in terms of NMI and F₁-score. Thus, we run HoKT with different orders to select the most suitable one and take the Cell Phone Call dataset as an example. We choose several typical snapshots to analyze the impact of different weight combinations, including t = 3, 4, 6, and 10.

We can find the great impact of different weight combinations on the algorithm’s performance. We illustrate the weight selection strategy based on these cases. At time step 3, second-order knowledge can be used. The results are shown in Table X and Fig. 16(a). The NMI and F₁-score found by second-order knowledge are higher than those obtained using first-order knowledge. This case means that HoKT using higher order knowledge provides a more structured division. How to determine the weight? The overlapping ratio r₁3 of snapshots between t = 1 and t = 3 is 0.7506 and r₂3 = 0.7600. The difference between those two r values is noticeable, and r₂3 is greater than r₁3. Thus, the weight for t = 2 should be greater than that for t = 1. We test four combinations for w and find that w = {0.6, 0.4} and w = {0.8, 0.2} are better than others.

The experimental results of different orders at time step 4 are shown in Table XI and Fig. 16(b). The overlapping ratio r₃4 of snapshots between t = 4 and t = 3 is 0.7555, r₂₄ = 0.7540, and r₁₄ = 0.7568. The difference between them is slight and only 0.0015. Interestingly, r₄₃ has the highest proportion, so the proportion of weights should be the highest for the first snapshot and the lowest for the second snapshot. If the second-order knowledge is used, the weight for t = 3 should

TABLE X
NMI AND F₁-SCORE WITH DIFFERENT ORDERS WHEN t = 3

| order | w | NMI | F₁-score |
|-------|---|-----|----------|
| 1     | 1 | 0.700±0.04 | 0.717±0.04 |
| 2     | (0.6, 0.4) | 0.744±0.05 | 0.747±0.04 |
|       | (0.7, 0.3) | 0.732±0.05 | 0.725±0.05 |
|       | (0.8, 0.2) | 0.745±0.05 | 0.745±0.05 |
|       | (0.9, 0.1) | 0.729±0.04 | 0.726±0.04 |

TABLE XI
NMI AND F₁-SCORE WITH DIFFERENT ORDERS WHEN t = 4

| order | w | NMI | F₁-score |
|-------|---|-----|----------|
| 1     | (1) | 0.539±0.07 | 0.538±0.07 |
| 2     | (0.6, 0.4) | 0.545±0.05 | 0.531±0.06 |
|       | (0.7, 0.3) | 0.509±0.10 | 0.502±0.10 |
|       | (0.8, 0.2) | 0.512±0.07 | 0.502±0.08 |
|       | (0.9, 0.1) | 0.541±0.07 | 0.531±0.10 |
| 3     | (0.5, 0.3, 0.2) | 0.526±0.01 | 0.511±0.01 |
|       | (0.6, 0.3, 0.1) | 0.478±0.09 | 0.463±0.05 |
|       | (0.7, 0.2, 0.1) | 0.489±0.00 | 0.487±0.01 |
|       | {0.2, 0.3, 0.5} | 0.551±0.03 | 0.543±0.04 |
TABLE XII
NMI AND F1-SCORE WITH DIFFERENT ORDERS WHEN t = 6

| order | w                  | NMI   | F1-score |
|-------|--------------------|-------|----------|
| 1     | (0.8, 0.4)         | 0.703 ± 0.06 | 0.690 ± 0.08 |
| 2     | (0.7, 0.3)         | 0.713 ± 0.05 | 0.702 ± 0.08 |
| 3     | (0.6, 0.3, 0.2)    | 0.777 ± 0.04 | 0.733 ± 0.06 |
| 4     | (0.6, 0.3, 0.1)    | 0.737 ± 0.07 | 0.733 ± 0.07 |
| 5     | (0.7, 0.2, 0.1)    | 0.739 ± 0.00 | 0.732 ± 0.00 |

TABLE XIII
NMI AND F1-SCORE WITH DIFFERENT ORDERS WHEN t = 10

| order | w                  | NMI   | F1-score |
|-------|--------------------|-------|----------|
| 1     | (0.5, 0.3)         | 0.691 ± 0.00 | 0.689 ± 0.00 |
| 2     | (0.6, 0.3, 0.1)    | 0.737 ± 0.07 | 0.733 ± 0.07 |
| 3     | (0.7, 0.2, 0.1)    | 0.739 ± 0.00 | 0.732 ± 0.00 |
| 4     | (0.4, 0.3, 0.2, 0.1) | 0.687 ± 0.05 | 0.657 ± 0.10 |
| 5     | (0.5, 0.3, 0.15, 0.05) | 0.704 ± 0.05 | 0.704 ± 0.05 |
| 6     | (0.6, 0.2, 0.15, 0.05) | 0.682 ± 0.05 | 0.667 ± 0.07 |
| 7     | (0.7, 0.2, 0.09, 0.01) | 0.708 ± 0.04 | 0.708 ± 0.04 |
| 8     | (0.5, 0.2, 0.15, 0.01) | 0.727 ± 0.07 | 0.727 ± 0.06 |
| 9     | (0.6, 0.2, 0.01, 0.09, 0.01) | 0.718 ± 0.05 | 0.719 ± 0.05 |

The experimental results of different orders at time step 6 are shown in Table XII and Fig. 16(c). The network at t = 6 is more similar to that at t = 5 and 4 than at t = 3, 2, and 1. Thus, second-order knowledge is only used. The weight for the fifth snapshot is greater than the fourth and third snapshots. We find HoKT achieves the best performance with w = {0.7, 0.2, 0.1} or w = {0.6, 0.3, 0.1}. We also find that third-order knowledge is better than second-order knowledge. Although the weight of the third snapshot is only 0.1, it can bring a big improvement.

The experimental results of different orders at time step 10 are shown in Table XIII and Fig. 16(d). At time step 10, r9, 10, r8, 10, and r7, 10 rank top three, and r9, 10 = r8, 10. Thus, if we select second-order knowledge and w = {0.5, 0.5} whose performance is similar to that of w = {0.7, 0.3}. If we select third-order knowledge and w = {0.6, 0.3, 0.1}. The great overlapping ratio illustrates more useful knowledge in this snapshot for the next snapshot. Although the results with higher order knowledge are not optimal, the bias is small, and a relatively satisfactory solution can be obtained, which may be attributed to the ensemble idea in the proposed scheme. Although we obtain the best performance with second-order knowledge, the performance gap between different combinations is large (large bias). Therefore, for new datasets, it is recommended to select slightly higher order knowledge.

V. CONCLUSION
HoKT has first gravitated to DCD problems over networks with great changes, which is essential in many fields. The proposed higher order knowledge transfer strategy with HONMI can balance the positive and negative transfer caused by great changes among snapshots. HoKT adapts to various scenarios, including network sizes, similarity distributions, and minor or great changes between snapshots. In the case of great changes, the performance improvement of HoKT is noticeable, and we still have a good performance in the case of minor changes. Moreover, if the overlapping rate is high (>0.95), the gain brought by higher order knowledge is not apparent but positive because the first-order knowledge already contains rich information.

The concept of high-order knowledge transfer can be extended to other dynamic optimization problems with great changes, not limited to the current community detection problem, providing a new solution. Also, the parameter w needs to be set by users in our algorithm. In the future, we can design a self-adaptive algorithm to obtain a better w.

APPENDIX
Four datasets are generated based on the four different kinds of events. Each network consists of 1000 nodes whose mean degree is 15. Table XIV depicts NMI on four networks in terms of three different algorithms. Table XV shows F1-score in terms of HoKT, DYNMOGA, and KT-MOEA/D. When t = 2, the first-order knowledge is used. When t = 3, the second-order knowledge is used and w = {0.8, 0.2}. When t = 4, the third-order knowledge is used and w = {0.8, 0.2, 0.1}. When t = 5, the second-order knowledge is used and w = {0.7, 0.3}. The gain brought by higher order knowledge is not obvious but positive because the first-order knowledge already contains rich information.

Fig. 17. Running time of HoKT for all datasets.
for all datasets is shown in Fig. 17 in the supplementary material. As the size of the network continues to increase, the algorithm’s runtime becomes longer.

REFERENCES

[1] S. Fortunato, “Community detection in graphs,” *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, 2010.

[2] V. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos, “Community detection in social media,” *Data Min. Knowl. Disc.* vol. 24, pp. 515–554, May 2012.

[3] S. Fortunato and D. Hric, “Community detection in networks: A user guide,” *Phys. Rep.*, vol. 659, pp. 1–44, Nov. 2016.

[4] A. Arenas, L. Danon, A. Díaz-Guilera, P. M. Gleiser, and R. Guimerà, “Community analysis in social networks,” *Eur. Phys. J. B.*, vol. 38, no. 3, pp. 373–380, 2004.

[5] H. Mahmoud, F. Masulli, S. Rovetta, and G. Russo, “Community detection in protein-protein interaction networks using spectral and graph approaches,” *Comput. Intell. Methods Bioinform. Biostat.*, pp. 62–75, Jan. 2014.

[6] S. Jia et al., “Defining and identifying cograph communities in complex networks,” *New J. Phys.*, vol. 17, no. 1, 2015, Art. no. 013044.

[7] Y. Li, K. He, D. Bindel, and J. E. Hopcroft, “Uncovering the small community structure in large networks: A local spectral approach,” in *Proc. WWW*, 2015, pp. 658–668.

[8] F. Liu et al., “Deep learning for community detection: Progress, challenges and opportunities,” in *Proc. IJCAI*, 2020, pp. 4981–4987.

[9] T. Aynaud and J.-L. Guillaume, “Static community detection algorithms for evolving networks,” in *Proc. IEEE 8th Int. Symp. Model. Optim. Mobile Ad Hoc Wireless Netw.*, 2010, pp. 513–519.

[10] F. Folino and C. Pizzuti, “An evolutionary multi-objective approach for community discovery in dynamic networks,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1838–1852, Aug. 2014.

[11] G. Rossetti and R. Cazabet, “Community discovery in dynamic networks: A survey,” *ACM Comput. Surveys*, vol. 51, no. 35, pp. 1–37, 2019.

[12] S. Asur, S. Parthasarathy, and D. Ucar, “An event-based framework for characterizing the evolutionary behavior of interaction graphs,” *ACM Trans. Knowl. Disc. Data*, vol. 3, no. 4, pp. 1–36, 2009.

[13] M.-S. Kim and J. W. Han, “A particle-and-density based evolutionary clustering method for dynamic networks,” *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 622–633, 2009.

[14] Y.-R. Lin, Y. Chi, S. H. Zhu, H. Sundaram, and B. L. Tseng, “Analyzing communities and their evolutions in dynamic social networks,” *ACM Trans. Knowl. Disc. Data*, vol. 3, no. 2, pp. 1–31, 2009.

[15] Y. Chi, X. D. Song, D. Y. Zhou, K. Hino, and B. L. Tseng, “Evolutionary spectral clustering by incorporating temporal smoothness,” in *Proc. KDD*, 2007, pp. 153–162.

[16] X. Zeng, W. Wang, C. Chen, and G. G. Yen, “A consensus community-based particle swarm optimization for dynamic community detection,” *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2502–2513, Jun. 2020.

[17] X. Ni, W. Li, and C. Q. Wu, “A label-based evolutionary computing approach to dynamic community detection,” *Comput. Commun.*, vol. 108, pp. 110–122, Aug. 2017.

[18] X. Ma and D. Dong, “Evolutionary nonnegative matrix factorization algorithms for community detection in dynamic networks,” *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1045–1058, May 2017.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multi-objective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Jun. 2002.

[20] A. Mirshahvalad, J. Lindholm, M. Derlén, and M. Rosvall, “Significant communities in large sparse networks,” *PLoS ONE*, vol. 7, no. 3, pp. 1–12, 2012.

[21] F. Liu, J. Liu, S. Xu, C. Zhou, I. Yang, and Q. Z. Sheng, “Detecting the evolving community structure in dynamic social networks,” *World Wide Web*, vol. 23, pp. 715–733, 2020.

[22] L. Yuan, Q. Zhu, Y. Zheng, W. Dong, Y. Ke, and Z. Li, “Temporal smoothness framework: Analyzing and exploring evolutionary transition behavior in dynamic networks,” in *Proc. IEEE 33rd Int. Conf. Tools Artif. Intell.*, 2021, pp. 1206–1210.

[23] C. Gao, Z. Chen, X. Li, Z. Tian, S. Li, and Z. Wang, “Multi-objective discrete particle swarm optimization for community detection in dynamic networks,” *Evrophys. Lett.*, vol. 122, no. 2, 2018, Art. no. 28001.

[24] K. C. Tan, L. Feng, and M. Jiang, “Evolutionary transfer optimization—A New Frontier in evolutionary optimization research,” *IEEE Comput. Intell. Mag.*, vol. 16, no. 1, pp. 22–23, Feb. 2021.

[25] M. Jiang, Z. Huang, L. Qiu, W. Z. Huang, and G. G. Yen, “Transfer learning based dynamic multi-objective optimization algorithms,” *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 501–514, Aug. 2018.

[26] M. Jiang, Z. Z. Wang, H. K. Hong, and G. G. Yen, “Knee point based imbalanced transfer learning for dynamic multi-objective,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 1, pp. 117–129, Feb. 2021.

[27] M. Jiang, Z. Z. Wang, L. Qiu, S. H. Guo, X. Gao, and K. C. Tan, “A fast dynamic evolutionary multi-objective algorithm via manifold transfer learning,” *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3417–3428, Jul. 2021.

[28] J. Liu, Q. Zhao, B. Yan, S. Elsayed, and R. Sarker, “Transfer learning-assisted multi-objective evolutionary clustering framework with decom-position for high-dimensional data,” *Inf. Sci.*, vol. 505, pp. 440–456, Dec. 2019.

[29] H. Zhang, G.-G. Wang, “Improved NSGA-III using transfer learning and centroid distance for dynamic multi-objective optimization,” *Complex Intell. Syst.*, to be published, doi: 10.1007/s40747-021-00570-z.

[30] Z. Liu and H. Wang, “Improved population prediction strategy for dynamic multi-objective optimization algorithms using transfer learning,” in *Proc. IEEE Congr. Evol. Comput.*, 2021, pp. 103–110.
[31] J. Li, T. Sun, Q. Lin, M. Jiang, and K. C. Tan, “Reducing negative transfer learning via clustering for dynamic multi-objective optimization,” IEEE Trans. Evol. Comput., vol. 26, no. 5, pp. 1102–1116, Oct. 2022.

[32] M. Jiang, Z. Wang, S. Guo, X. Gao, and K. C. Tan, “Individual-based transfer learning for dynamic multi-objective optimization,” IEEE Trans. Cybern., vol. 51, no. 10, pp. 4968–4981, Oct. 2021.

[33] D. Xu, M. Jiang, W. Hu, S. Li, R. Pan, and G. G. Yen, “An online prediction approach based on incremental support vector machine for dynamic multi-objective optimization,” IEEE Trans. Evol. Comput., vol. 26, no. 4, pp. 690–703, Aug. 2022.

[34] W. Zhou, L. Feng, K. C. Tan, M. Jiang, and Y. Liu, “Evolutionary search with multi-view prediction for dynamic multi-objective optimization,” IEEE Trans. Evol. Comput., vol. 26, no. 5, pp. 911–925, Oct. 2022.

[35] Z. Wang, M. Jiang, X. Gao, L. Feng, W. Hu, and K. C. Tan, “Evolutionary dynamic multi-objective optimization via regression transfer learning,” in Proc. IEEE Symp. Comput. Intell., 2019, Art. no. 19379166.

[36] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, “Transfer learning for gaussian process assisted evolutionary bi-objective optimization for objectives with different evaluation times,” in Proc. GECCO, 2020, pp. 587–594.

[37] C. Pizzuti, “Evolution computation for community detection in networks: A review,” IEEE Trans. Evol. Comput., vol. 22, no. 3, pp. 464–483, Jun. 2018.

[38] Z. Dhouioui and J. Akaichi, “Tracking dynamic community evolution in social networks,” in Proc. IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. Min., 2014, pp. 764–770.

[39] N. P. Nguyen, T. N. Dinh, Y. Xuan, and M. T. Thai, “Adaptive algorithms for detecting community structure in dynamic social networks,” in Proc. IEEE INFOCOM, 2011, pp. 2282–2290.

[40] Y. Xin, Z. Q. Xie, and J. Yang, “An adaptive random walk sampling method on dynamic community detection,” Exp. Syst. Appl., vol. 58, pp. 10–19, Oct. 2016.

[41] G. Rossetti, L. Pappalardo, D. Pedrechi, and F. Giannotti, “Tiles: An online algorithm for community discovery in dynamic social networks,” Mach. Learn., vol. 106, no. 8, pp. 1213–1241, 2017.

[42] T. B. Wolf, C. Tantipathananandh, and D. Kempe, “Dynamic community identification,” in Models, Algorithms, and Applications. New York, NY, USA: Springer, 2010, pp. 307–336.

[43] N. Ilhan and Ş. G. Ögüdüzü, “Predicting community evolution based on time series modeling,” in Proc. IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. Min. (ASONAM), 2015, pp. 1509–1516.

[44] Y. Yin, Y. Zhao, H. Li, and X. Dong, “Multi-objective evolutionary clustering for large-scale dynamic community detection,” Inf. Sci., vol. 549, pp. 269–287, Mar. 2021.

[45] Z. Wang, C. Wang, C. Gao, X. Li, and X. Li “An evolutionary autoencoder for dynamic community detection,” Sci. China Inf. Sci., vol. 63, Sep. 2020, Art. no. 212205.

[46] C.D. Wang, J. H. Lai, and P. S. Yu, “Dynamic community detection in weighted graph streams,” in Proc. SIAM Int. Conf. Data Min. (SDM), 2013, pp. 764–770.

[47] X. Huang, H. Cheng, L. Qin, W. Tian, and J. Yu, “Querying k-truss community in large and dynamic graphs,” in Proc. SIGMOD, 2014, pp. 1311–1322.

[48] M. Gong, L. Jiao, D. Yang, and W. Ma, “Evolutionary multi-objective optimization algorithms,” J. Softw., vol. 20, no. 2, pp. 271–289, 2009.

[49] N. Srinivas and K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms,” Evol. Comput., vol. 2, no. 3, pp. 221–248, 1994.

[50] Y. Censor, “Pareto optimality in multi-objective problems,” Appl. Math. Optim., vol. 4, pp. 41–59, Mar. 1977.

[51] M. D. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” Phys. Rev. E Stat. Nonlinear, Biol. Soft Matter Phys., vol. 69, Feb. 2004, Art. no. 026113.

[52] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, “FacetNet: A framework for analyzing communities and their evolutions in dynamic networks,” in Proc. 17th Int. Conf. World Wide Web, 2008, pp. 685–694.

[53] P. Rajhans and D. B. Skillcorn, “Structure in the enron email dataset,” Comput. Math. Org. Theory, vol. 11, pp. 183–199, Oct. 2005.

[54] D. Duan, Y. Li, Y. Jin, and Z. Lu, “Community mining on dynamic weighted directed graphs,” in Proc. 1st Int. Workshop Complex Netw. Meet Inf. Knowl. Manag., 2009, pp. 11–18.

[55] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, “Task relatedness based multitask genetic programming for dynamic flexible job shop scheduling,” IEEE Trans. Evol. Comput., early access, Aug. 18, 2022, doi: 10.1109/TEVC.2022.3199783.