Dynamic generation of internet of things organizational structures through evolutionary computing

Shen, Zhiqi; Yu, Han; Yu, Ling; Miao, Chunyan; Chen, Yiqiang; Lesser, Victor R.

2018

Shen, Z., Yu, H., Yu, L., Miao, C., Chen, Y., & Lesser, V. R. (2018). Dynamic generation of internet of things organizational structures through evolutionary computing. IEEE Internet of Things Journal, 5(2), 943-954. doi:10.1109/JIOT.2018.2795548

<https://hdl.handle.net/10356/90165>

<https://doi.org/10.1109/JIOT.2018.2795548>

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at:
<https://doi.org/10.1109/JIOT.2018.2795548>

Downloaded on 26 Apr 2023 21:42:14 SGT

Dynamic Generation of Internet of Things Organizational Structures through Evolutionary Computing

Zhiqi Shen, *Member, IEEE*, Han Yu, Ling Yu, Chunyan Miao, Yiqiang Chen and Victor R. Lesser, *Fellow, IEEE*

Abstract—In today's world, intelligent embedded devices and sensors are interconnected into a dynamic and global network infrastructure is referred to as the Internet-of-Things (IoT). It has been widely recognized that the performance of an IoT is highly affected by how it is organized. A large scale system may have billions of possible ways of being organized, which makes it impractical to find a high quality choice of organization by manual means. In this paper, we propose a genetic algorithm aided framework for generating hierarchical IoT organizational structures. We propose a novel unique mapping between organizational structures and genome representations. Since hierarchical (i.e. tree-structured) organizations are one of the most common forms of organizations, we propose a novel method to map the phenotypic hierarchical structure space into a genome-like array representation space. This new representation opens up opportunities for evolutionary computing techniques to help IoT applications automatically generate organizational structures according to desired objective functions. Based on this mapping, we introduce the *hierarchical genetic algorithm* (HGA) which enriches standard genetic programming approaches with the hierarchical crossover operator with a repair strategy and the mutation of small perturbation operator. The proposed approach is evaluated in an IoT-based information retrieval system. The results have shown that competitive baseline structures which lead to IoT organizations with good performance in terms of utility can be found by the proposed approach during the evolutionary search. Compared with the traditional genetic operators, the newly introduced operators produced organizations of higher utility more consistently under a variety of test cases. The proposed approach is computationally efficient in large search spaces and provides a novel method for future generations of IoT systems to autonomously improve performance.

Index Terms—Internet-of-Things (IoT), organization, evolutionary computation

1 INTRODUCTION

THE Internet-of-Things (IoT) represents a paradigm shift in today's global wireless communication infrastructure [1], [2]. Digital devices such as Radio-Frequency IDentification (RFID) tags, sensors, actuators and data storage and processing devices which have varying degrees of self-configuration and can interact with each other and cooperate with their neighbors to address certain problems. In cities such as Singapore and New York, IoT has permeated many aspects of urban living [3], enabling the emergence of smart cities with a “digital nervous system” [4], [5] for efficient planning and management.

The classical definition of IoT focuses on the infrastructural aspect of connecting communication devices to enable

information flow [6]. As more smart computing devices such as smartphones equipped with significant computing power and data processing software operated by human users become part of the IoT, collaboration network as part of the IoT infrastructure may dynamically emerge [7], [8], [9]. These smart devices coupled with human behaviors are increasingly steering IoTs towards Internet-of-Agents (IoA) [10] and push computing to the “edge” of IoT [11]. Future generations of IoT systems will need to rely on organizational structures resembling those found in social organizations [12], [13], [14], [15], [16] in order to stay efficient in the face of these new collectives of smart devices with human in the loop [17].

The organization of a cooperative IoT components (which collaborate in order to achieve common system objectives) can provide a framework for activities and interactions through the definition of roles, groups, tasks, interaction norms and authority relationships [18]. In an organizational context, an IoT *node* is an active and communicating entity playing roles within groups. In the extended IoA scenarios, it can be an artificial automated entity (such as a webservice) or a human being. A *group* is a set of IoT nodes sharing certain traits. It is commonly used to partition an organization. A *role* is the abstract representation of a functional position for an IoT node in a group. A node must play at least one role in an organization. Otherwise, it becomes redundant and reduces the efficiency of the

- Z. Shen and L. Yu are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798.
E-mail: zqshen@ntu.edu.sg
- H. Yu and C. Miao are with the Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), Nanyang Technological University, Singapore 639798.
E-mail: han.yu@ntu.edu.sg, ascymiao@ntu.edu.sg
- Yiqiang Chen is with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China; the University of Chinese Academy of Sciences, Beijing, China; and the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Beijing, China.
E-mail: yqchen@ict.ac.cn
- V. R. Lesser is with the School of Computer Science, University of Massachusetts Amherst, Amherst, MA 01003, USA.

Manuscript received June 23, 2017; revised Nov 2017

organization.

Borrowing similar concepts from multi-agent systems [19], [20], [21], [22], in large scale IoT systems, to form and evolve an organization makes it possible for the system to exploit the collective efficiency of the nodes and manage emerging situations. This is reflected from the DARPA Network Challenge in which people dynamically form online teams to play the role of propagating information about and/or physically searching for 10 red weather balloons placed at locations around the continental United States [23]. Experiments in similar application domains have shown that various organizations with the same set of entities have different impact on its overall performance [24].

Among all organization phenotypes, the hierarchical structure is one of the most common structures observed [25]. Many real-world organizations can be abstracted as hierarchical, tree-like structures or sets of parallel hierarchical structures, where entities are categorized in different levels in the hierarchies [25]. Often, the level at which an entity resides indicates its capabilities and roles. A specific level in the system often consists of equally capable entities, performing similar roles, as seen in the distributed information retrieval (IR) system described in [24].

As the IoT field moves forward, the problem of how to organize IoT components into efficient structures to support dynamic collaboration becomes increasingly important. For IoTs with a large number of components, there exists a vast variety of possible ways to organize them. Due to the differences in the depth and the width of a hierarchical structure, the number of organization instances increases exponentially with the number of nodes. There lacks an effective mechanism to search for organization instance for a given set of IoT nodes that can achieve satisfactory expected system performance. To address the challenge of dynamic organization of IoT nodes, we propose a genetic algorithm (GA) approach in this paper, namely the *hierarchical genetic algorithm (HGA)*. Since hierarchical (i.e. tree-structured) organizations are one of the most common forms of organizations, we propose a novel method to map the phenotypic hierarchical structure space into a genome-like array representation space. This new representation opens up opportunities for evolutionary computing techniques to help IoT applications automatically generate organizational structures according to desired objective functions. HGA enriches standard genetic programming approaches with the hierarchical crossover operator with a repair strategy and the mutation of small perturbation operator. HGA serves as an alternative to the conventional enumeration methods for building hierarchical organizations to locate appropriate organization instances in a vast search space. We assume the existence of a coordination entities for a given group of IoT nodes which is responsible for executing the HGA when new nodes join the group or when other changes in the group occur. HGA is evaluated in a real-world information retrieval application scenario involving multiple sensors and aggregators [24] which exhibits numerous possible organizational variants. The organizations obtained by HGA significantly outperform those obtained by alternative approaches in terms of efficiency.

The rest of this paper is organized as follows: In Section 2, we review several major organization design methods

as well as important evolutionary approaches. In Section 3, we propose a novel genome representation of organizations supporting the proposed evolutionary IoT organization generation approach. In Section 4, we propose the enhanced the crossover and mutation operators for the genome representation of organizations. Section 5 evaluates the proposed approach in information retrieval applications and demonstrates its advantages through comparisons with state-of-the-art existing approaches. Section 6 concludes the paper and discusses potential future research directions.

2 RELATED WORK

To the best of our knowledge, this paper is the first to propose a computational approach to dynamically generate organizational structures in IoT. As the research of organizational structure is interdisciplinary in nature, we review related in the similar domain of multi-agent system (MAS) organization research to discuss how the proposed approach advances the collective literature of dynamically organizing diverse entities at scale.

MAS organization has been investigated by many researchers. Early methodologies such as Gaia [26] and OMNI [27] aim to provide guidelines to assist the manual MAS organization design process. Nevertheless, the manual design process is time consuming, depends heavily on the expertise of the designers, and is infeasible in large-scale IoT systems in which entities can dynamically join or leave the system. Thus, it is desirable to automate the process of organization.

In order to automate the design of organizational structures, quantitative metrics are needed to rapidly and precisely assess the expected performance of a given organizational structure. In [24], an organizational design modeling language (ODML) is proposed. The utility function for an organization is defined in the context of quantitative measurement of the performance of a distributed sensor network and an information retrieval system typical of today's IoT systems. The exploitation of hard constraints and equivalence classes, parallel search, and the use of abstractions, have been studied in this study to help designers reduce the complexity of coming up with a valid organizational structure. Nevertheless, this process remains largely manual even with ODML.

Another MAS organization designer, KB-ORG [28], uses both application-level and coordination-level organization design knowledge to explore the search space of candidate organizations selectively. This approach significantly reduces the exploration effort required to produce effective designs as compared to modeling and evaluation-based approaches that do not incorporate designer expertise. Nonetheless, similar to ODML, KB-ORG only aims to help designers prune the search space. The designers still have to manually come up with candidate organizational structures.

Genetic algorithms (GA) have started to show its promise as an alternative to exhaustive methods such as breadth-first search and depth-first search for dynamically generating organizational structures in IoT systems. In [29], [30], [31], we proposed a basic framework for a GA-based heuristic search method for optimizing organizational structures. In this paper, we propose a key enabling advance – a unique one-to-one mapping between organizational

structure representations and genome representations. This advance, coupled with the enhanced operators, enables GA to be applied to dynamically generate useful organizational structures for IoT systems.

3 A GENOME REPRESENTATION OF ORGANIZATIONAL STRUCTURES

The organization of a hierarchy consists of a number of tree structures. According to the number of “leaders”, it can be either a single tree or a set of trees. The intermediate nodes in a tree are responsible for assigning tasks to their subordinates, as well as reporting results back to their superiors. There is little interaction among nodes on the same level, or among different hierarchies. The leaf nodes are at the bottom of the structure. They are responsible for completing the most basic tasks.

Optimization in such a search space can be handled by evolutionary algorithms [32], especially genetic programming, which supports populations of tree structures. It has also been shown that some well-structured trees (e.g. binary trees), with a certain number of levels and a fixed number of subordinates per node, can be represented by arrays [33], [34]. Transformations are feasible as a result of their regular structures. This allows the traditional crossover and mutation operators of other evolutionary algorithms, such as genetic algorithms, to take effect.

We propose a novel array representation of hierarchical organizational structures which is applicable to a much broader range of hierarchical structures than just binary trees. It converts a set of hierarchical trees into a fixed-length array with integer components, which resemble gene sequences. The representation is not limited to describing a single tree, and the number of subordinates of each node is not limited by a constant value. Unbalanced trees, in which leaf nodes are not on the same hierarchical level, can also be described using this representation. This allows the traditional crossover and mutation operators of evolutionary algorithms to take effect.

We assume that the target hierarchical organizations considered here have the following properties:

- 1) *Assumption 1*: The number of leaf nodes is fixed.
- 2) *Assumption 2*: The maximum possible number of levels is fixed.

These assumptions imply that the total number of nodes in the organization is bounded. Based on these assumptions, we can make use of the partition concept to convert the organization from tree structures to arrays.

Let N be the total number of leaf nodes (or end nodes), so that they can be numbered as $1, 2, \dots, N$ respectively from left to right. Let M be the maximum tree depth (i.e. maximum height of the structure). The reason for limiting the height is that very tall structures can be slow or irresponsive, as the long path length from root to leaf increases message latency among the nodes. The organization of a hierarchical IoT system can be represented as $[a_1, a_2, a_3, \dots, a_{N-1}]$, where a_i is an integer between 1 and M , expressing the level number where leaf nodes i and $i+1$ start to separate. We denote this form of organizational structure representation as *Representation 1*.

3.1 Analysis

Theorem 1. The proposed organization representation has the following properties:

- 1) For every hierarchical organization instance which satisfies Assumptions 1 and 2, the array representation that can be generated is unique.
- 2) There exists an organization instance for every array representation in the proposed form.

Proof 1. We first prove the existence of an array representation for every hierarchical organization instance. The way of generating an array representation of an arbitrary hierarchical organization instance can be expressed as follows. If there are N leaf nodes, we prepare $N-1$ slots. Firstly, organize the structure well so that the root nodes, intermediate nodes, and leaf nodes are on their proper designated levels. Secondly, we examine the separation pattern between adjacent leaf nodes one by one from left to right. Fill the slots with the level number where the adjacent leaf nodes start to separate. Using Figure 1 as an example, the first two leaf nodes on the left are direct subordinates of the first tree root (i.e. on the root level (Level 1)). They do not separate. However, on Level 2, they separate into different nodes. Thus, the first number is 2. The second slot should also be filled with 2 because the second and third leaf nodes on the left separate on Level 2. As the third and fourth leaf nodes are direct subordinates of an intermediate node on Level 2, they start to separate on Level 3. Therefore, “3” should be the third number in the array. Following this process, we can obtain the values, which are the level numbers, for all the slots. Together they form the proposed representation. In this way, the proposed genome representation ensures that *there exists an array representation for every hierarchical organization instance*.

We then prove the uniqueness of the generated array representation. If array representations $[a_1, a_2, a_3, \dots, a_{N-1}]$ and $[b_1, b_2, b_3, \dots, b_{N-1}]$ which are derived from the same organization instance are different, there exists an $i \in 1, 2, \dots, N$ such that $a_i \neq b_i$. This shows that the leaf nodes i and $i+1$ separate at different levels in the two corresponding organization structures. This means the organization structures are not identical, which contradicts the pre-condition that $[a_1, a_2, a_3, \dots, a_{N-1}]$ and $[b_1, b_2, b_3, \dots, b_{N-1}]$ are derived from the same MAS organization instance. In this way, the proposed genome representation ensures that *each genome representation corresponds to a unique hierarchical organization instance*.

Proof 2. Given an array representation with positive integers of length L , we would like to construct an organization instance containing $L+1$ leaf nodes as follows. Find all the digits “1” in the representation (if there are any). Calculate the number of digits (greater than 1) between adjacent “1”s one by one from left to right, and denote them as $n_1, n_2, n_3, \dots, n_{k+1}$, where k is the number of “1”s. If there are no “1”s, then $k=0$ and $n_1=L$. The corresponding organization has $k+1$ root nodes with $n_1+1, n_2+1, n_3+1, \dots, n_{k+1}+1$ leaf nodes, respectively, from left to right. So far we have completed the root level

(Level 1) of the organization. For instance, with array $[2, 2, 3, 1, 2, 3]$, $n_1 = 3$, $n_2 = 2$, i.e. there are two root nodes with 4 and 3 leaf nodes respectively. For Level 2, we take segments with "1"s and "2"s as separators. These segments should only contain digits greater than 2 (if any). Like what is done for Level 1, the number of digits between adjacent separators are recorded as $r_1, r_2, r_3, \dots, r_{t+1}$, where t is the total number of "1"s and "2"s. If $r_i = 0$, it corresponds to a leaf node; otherwise, it corresponds to an intermediate node on Level 2. After that, take segments with "1"s, "2"s, and "3"s as separators, and repeat the steps until the greatest numbers in the representation are examined. In this way we can obtain the full organization instance.

3.2 Example Genome Representation

An example with seven leaf nodes ($N = 7$) is illustrated in Figure 1. It consists of two trees. On Level 1, the four leaf nodes on the left and the three leaf nodes on the right separate into two trees. In other words, there is a separation between the leaf nodes 4 and 5, so $a_4 = 1$. On Level 2, there are two leaf nodes and one intermediate node (three nodes altogether) under the left tree root, corresponding to the $[2, 2]$ (two partition numbers) to the left of "1" in the array. The one leaf node and one intermediate node (two nodes altogether) under the right tree root are represented by the "2" (one partition number) to the right of "1". Both intermediate nodes on Level 2 have two leaf nodes as their subordinates (leaf nodes 3 and 4, leaf nodes 6 and 7), which are separated on Level 3, resulting in the two "3"s at the 3rd and 6th digits in the array. Therefore, we obtain the genome array representation for the organization as $[2, 2, 3, 1, 2, 3]$.

Fig. 1. A sample organization which can be represented as $[2, 2, 3, 1, 2, 3]$. Nodes are displayed as circles in the figure. Leaf nodes are numbered.

Conversely, we can also obtain an organizational structure by interpreting the proposed genome representation. For instance, if we want to determine which level Node 4 sits on, we need to examine both the node's left and right neighbors. The 3rd and 4th digits in the array are "3" and "1". It means that Node 3 and Node 4 are separated on Level 3. Node 4 and Node 5 are separated on Level 1. As a result, Node 4 is on Level 3 (the larger number between 1 and 3). Similarly, because the 5th digit is "2", i.e. Node 5 and Node 6 are separated on Level 2, Node 5 should be on Level 2 (the larger number between 1 and 2).

4 HIERARCHICAL GENETIC ALGORITHM (HGA)

4.1 Enhanced Crossover Operator

The traditional one-point crossover operator in GAs chooses a random slicing position along the chromosomes of two

(a) Genome representations for organizational structures. The sub-figure on the left shows the effect of one-point crossover. The sub-figure on the right shows the effect of hierarchical crossover with repair operation.

(b) The effect of one-point crossover on the two array representations on the left of Figure 2(a) illustrated in the form of organizational structures.

(c) The effect of hierarchical crossover with repair operation on the two array representations on the right of Figure 2(a) illustrated in the form of organizational structures.

Fig. 2. Illustration of one-point crossover and hierarchical crossover using *Representation 1* of the proposed genome representation of organizational structures.

parent solutions. All data beyond that point in either solution are swapped between the two parent solutions. The resulting chromosomes are two offspring solutions. Though commonly used, this crossover method only influences the structure near the crossover point, as shown in Figure 2(a) and Figure 2(b). However, it may not be enough to generate new offsprings in large-scale systems.

To speed up the evolution and increase the chance of obtaining a desirable structure with higher utility, new crossover operators are needed. Here, we propose an enhanced crossover operator – hierarchical crossover – specially designed for generating hierarchical organizations. The hierarchical crossover operator, based on *Representation 1*, contains swapping of sub-organizations and a repair strategy to keep the total number of leaf nodes constant.

It is implemented as follows.

Firstly, we compare the number of levels of two randomly selected organizations from the solution population. Denote the organization with more levels as the first solution and denote its number of levels as T . Denote the organization with fewer levels as the second solution. In case of ties, the order can be arbitrarily assigned. Secondly, select a node randomly from all nodes whose level number is between 1 and $T - 1$ from the first solution and denote the level number of the chosen node as S . Thirdly, select a node randomly at Level S or the penultimate level (whichever is smaller) from the second solution, and exchange the sub-structures between the two solutions beneath the selected nodes. If any of the solution candidates have only one level, generate two random solutions of the maximum tree depth as their offsprings instead. The exchange ensures that the two newly formed organizational structures do not exceed the maximum height of their parent structures. The detailed pseudo code of the hierarchical crossover operator is given in Algorithm 1.

Algorithm 1 Hierarchical crossover

Require: The array representations of two selected parents: p_1 and p_2 .

- 1: **if** $\max(p_1) < \max(p_2)$ **then**
- 2: Exchange p_1 and p_2 ;
- 3: **end if**
- 4: $T = \max(p_1)$;
- 5: **if** $T = 1$ **or** $\max(p_2) = 1$ **then**
- 6: Randomly generate two offsprings, o_1 and o_2 , of maximum tree depth;
- 7: **end if**
- 8: **return** o_1 and o_2 .

Nevertheless, the exchanged sub-structures do not necessarily contain equal number of leaf nodes. Thus, we propose the following repair strategy. Find the parent solution with the longer representation and randomly pick one digit from it and insert this digit into a random slot in the other parent solution. Continue until the two solutions have equal length. This will guarantee the validity of the two solutions, as shown in Figure 2(a) and Figure 2(c). Illustrated in both the array representation and the organization structures, Figure 2 displays the difference between the proposed hierarchical crossover and one-point crossover.

More specifically, randomly select a node from the above list as the crossover point cp_1 and denote the level number of cp_1 as S . Set ss_c1 to be the segments of the array representation of the sub-structure below cp_1 . Set ss_l1 to be the segments of the array representation to the left of the sub-structure below cp_1 and ss_r1 to be the segments of the array representation to the right of the sub-structure below cp_1 . Then, randomly select a node cp_2 from p_2 at the Level No. $\min[S, \max(p_2) - 1]$. Set ss_c2 to be the segments of the array representation of the sub-structure below cp_2 . Set ss_l2 to be the segments of the array representation to the left of the sub-structure below cp_2 and ss_r2 to be the segments of the array representation to the right of the sub-structure below cp_2 . Then, execute the genome repair strategy as detailed in Algorithm 2.

Algorithm 2 Genome repair strategy

Require: The list of all possible crossover nodes of p_1 from Level 1 till Level $T - 1$ after executing Algorithm 1.

- 1: $o_1 = [ss_l1, ss_c2, ss_r1]$ and $o_2 = [ss_l2, ss_c1, ss_r2]$;
- 2: **if** $\text{length}(o_1) > \text{length}(p_1)$ **then**
- 3: $\text{exnum} = \text{length}(o_1) - \text{length}(p_1)$;
- 4: **for** $j = 1 : \text{exnum}$ **do**
- 5: Randomly select an integer k_1 between 1 and $\text{length}(o_1)$;
- 6: Randomly select an integer k_2 between 1 and $\text{length}(o_2) + 1$;
- 7: $o_2 \leftarrow [o_2(1 : k_2 - 1), o_1(k_1), o_2(k_2 : \text{end})]$;
- 8: $o_1 \leftarrow [o_1(1 : k_1 - 1), o_1(k_1 + 1 : \text{end})]$;
- 9: **end for**
- 10: **else if** $\text{length}(o_2) > \text{length}(p_2)$ **then**
- 11: $\text{exnum} = \text{length}(o_2) - \text{length}(p_2)$;
- 12: **for** j from 1 to exnum **do**
- 13: Randomly select an integer k_2 between 1 and $\text{length}(o_2)$;
- 14: Randomly select an integer k_1 between 1 and $\text{length}(o_1) + 1$;
- 15: $o_1 \leftarrow [o_1(1 : k_1 - 1), o_2(k_2), o_1(k_1 : \text{end})]$;
- 16: $o_2 \leftarrow [o_2(1 : k_2 - 1), o_2(k_2 + 1 : \text{end})]$;
- 17: **end for**
- 18: **end if**
- 19: **return** o_1 and o_2 .

As observed from Figure 2, a branch of the tree corresponds to a piece of gene fragment. By swapping the two selected gene segments in the parents, we can obtain two new IoT organization instances with exchanged sub-organizations. This step is similar to two-point crossover, in which the segments between the two randomly selected crossover points of both parents are swapped to form the offsprings. However, like one-point crossover, two-point crossover also does not guarantee the selected gene segments correspond to whole tree branches. As long as the two crossover points are determined, the locations of the segments in the arrays do not change. Hierarchical crossover is different from two-point crossover in that it focuses on the branches of the tree structures and only change the gene segments that represent whole branches. Moreover, the locations of the two gene segments of the parents may differ from each other, and the repair strategy promotes

Algorithm 3 Mutation of small perturbation

Require: The array representation of an offspring created by the crossover operator (o_i); the length of this representation (numVar); the mutation probability (mutOps); and the maximum tree depth (maxTreeDepth).

- 1: $rN = \text{rand}(1, \text{numVar}) < \text{mutOps}$;
- 2: $o_i \leftarrow o_i + rN \times (\text{rand}(1, \text{numVar}) \times 2 - 1)$;
- 3: **if** $\text{length}(o_i) = 0$ **or** $\text{length}(o_i) > \text{maxTreeDepth}$ **then**
- 4: Retore o_i to its original value;
- 5: **end if**
- 6: **return** o_i .

organization update.

4.2 Enhanced Mutation Operator

In addition, we propose the mutation of small perturbation operator. The pseudo code of the enhanced mutation operator based on *Representation 1* is displayed in Algorithm 3. It is different from bit-wise mutation in that the digits can only increase by 1 or decrease by 1 with equal probabilities. If the perturbed digit is out of bounds, the operator restores it to its original value. In summary, the overall workflow of the proposed HGA is illustrated in Figure 3.

Fig. 3. The flowchart of the proposed algorithm.

5 EXPERIMENTAL EVALUATION

In this section, we demonstrate how HGA can be used to generate useful organizational structure for an IoT system designed to perform information retrieval (IR) tasks in a peer-to-peer (P2P) manner, and evaluate its performance during the process. The advantages of HGA in the context of such real-world applications are further discussed in comparison with well-known approaches including ODML [24] and KB-ORG [28].

5.1 Experiment Setup

To evaluate the effectiveness of the proposed approach, we will examine it in the context of the P2P IR systems. A generic model of organization for such a system is documented in [24]. Such a model is widely used in today's IoT distributed sensing and data fusion systems as well [35]. A P2P IR system, in general, is made up of a set of nodes connected in a P2P manner. Each node shares information with others. These nodes work in cooperation to provide IR services to users. Each node maintains an index and an IR search engine for its local information storage. A node in such a P2P IR system may perform a multitude of roles such as forwarding search queries, perform IR operations on its local information storage, or returning results to the search initiator. In such a fully distributed system, the efforts need to be organized in order to locate the results for the queries efficiently.

A structured, hierarchical organization composed of roles including mediators, aggregators, and databases is used to model the IR system in our experiment. A node is assigned to perform a corresponding role. The *information recall time* and the *query response time* are combined to form a single scalar metric to determine the utility of a generated organization instance. Detailed procedures to calculate the utility value can be found in [24].

As a general rule for organizational structure design in this type of applications, at the top level of each hierarchy is a mediator. The user sends a query, which will be assigned to a randomly selected mediator to handle. It uses the information catalogs of all the mediators to compare data sources, then routes the query to those mediators that appear able to locate the target information. After the query has been directed through the aggregators and processed by all the databases under the selected mediators, the responsible mediator will then collect and deliver the results.

Following the aforementioned general rule, Level 1 is the mediator level, where nodes are all mediators. The intermediate nodes correspond to aggregators, and the leaf nodes are databases. The whole organization can be outlined by a set of trees. In the experiment, we assume that all the databases in the system contain the same amount of topic data. Thus, these leaf nodes can be treated as homogeneous and *Representation 1* can be applied. From a practical viewpoint, we notice that it is not necessary to include an aggregator if it only has one subordinate, because it will increase the information transmission overhead and not bring any integration advantage. Hence, if such an organization instance emerges, we can simply omit the aggregator node and reduce the organization structure by one level.

Related modification can be made in the array representation, which is summarized below. Firstly, obtain all the segments of a genome between adjacent mediators (i.e. the integer series between "1"s). Set the smallest values of these segments to "2". Secondly, obtain all the segments with "1"s and "2"s as separators. Set the smallest values of these segments to "3". Continue until the highest level of the organization. Figure 4 shows the detailed steps of simplifying an example organization. It transforms a 5-level organization for an example P2P IR system to a 4-level one. The simplifying procedure is employed in the experiments

to achieve higher efficiency. At the same time, the number of organization instances we have to evaluate for every representation is reduced to one.

Fig. 4. Simplifying the genome representation of an organization. Nodes labelled "M" are mediators, nodes labelled "A" are aggregators, and nodes labelled "D" are databases.

According to [24], every mediator has got a rank according to its perceived response size. The one with the largest perceived response size receives rank No. 1, and the same rank is given to mediators with equal perceived response sizes. Mediators are chosen to be sent queries based on their ranks, resulting in the query probability $p_m, m \in (1, 2, \dots, N_m)$, where N_m is the total number mediator nodes in an organization. This is used to calculate the response recall, γ , of the organization, which is given by the following equation:

$$\gamma = \frac{\sum_{m=1}^{N_m} p_m \times \gamma_m}{\epsilon} \quad (1)$$

where γ_m is the actual response size of mediator m and ϵ represents the environment topic size as defined in [24]. The utility of organization is computed by combining the aspects of response recall and response time with appropriate weights of each term as follows:

$$u = 1000\gamma - \frac{t_r}{10} \quad (2)$$

where t_r is the response time as defined in [24]. The utility value serves as the *fitness* measure for each generated organization during the evolution.

We recognize that there are likely multiple suitable solutions that achieve the same utility in a given system, owing to the symmetry of the structures. Therefore, we need a method that allows growth in several promising areas in the search space. In other words, the diversity of the population should be enhanced and over-convergence should be avoided. We increase the competition between similar individuals by applying the restricted tournament selection (RTS) method described in [36]. It helps preserve diverse building blocks for locating appropriate organizations.

We examine the test cases with 12, 14, 16, 18, 20, 22, 24, 26, 28 and 30 databases (i.e. leaf nodes). The maximum height of the structures is set to be 4. The population size and the maximum number of candidate organizations used are shown in Table 1. All algorithms use a window size $w = 5$ for RTS in the population updating stage [36]. The

mutation rate is set to 10%. Each test case involves 10 independent runs.

TABLE 1
Experiment Configurations

Number of Leaf Nodes	Population Size	No. of Candidate Evaluations
12	50	2,000
14	100	5,000
16	200	10,000
18	500	50,000
20	500	50,000
22	500	50,000
24	500	100,000
26	500	100,000
28	500	100,000
30	1,000	200,000

The environment parameter settings in the experiment are as follows: message latency time = 20 milliseconds, process service rate = 10 per second, response service rate = 20 per second, and query rate = 3 per second. The search set size and query set size are set to be the total number of mediators for each organization. The response recall is therefore identical (100%) in all cases, and the utility is determined by the response time. The computation time of the genetic operators and population updating is negligible compared to that of the candidate evaluations.

5.2 Evaluation Metrics

We compare the proposed HGA with the standard genetic algorithm using one-point crossover with bit-wise mutation (SGA1) as well as the two-point crossover with bit-wise mutation (SGA2) in order to demonstrate the advantages of the newly introduced operators. We examine the performance of the approaches according to three metrics specific to the IR scenario: the *accuracy*, the *stability* of search and the *rate of convergence*:

- 1) The percentage relative error (PRE), e is calculated as:

$$e = \frac{f_{best} - f}{f_{best}} \quad (3)$$

where f_{best} is the best known fitness value among all the runs of all the algorithms for a given test case, and f is the current fitness value achieved. Then, $\bar{e}$ is the average of the PRE values among all the independent runs of each test case. The lower the value of $\bar{e}$, the better an organization instance performs.

- 2) The success rate s is calculated as:

$$s = \frac{n_{best}}{n} \quad (4)$$

where n_{best} the number of runs in which the best known solution is found and n is the total number of runs in each test case. The higher the value of s , the more stable a P2P IR system performs under a given organizational structure. Thus, a organization should preferably achieve low $\bar{e}$ and high s at the same time.

- 3) Last but not least, the rate of convergence of an approach is measured in the number of candidate organizational structures which must be evaluated before the fitness value stabilizes (i.e. a useful organizational structure has been generated).

5.3 Analysis of Results

Fig. 5. Experimental results comparing SGA1, SGA2 and HGA in terms of APRE and SR.

Figure 5(a) shows the $\bar{e}$ and s values achieved by SGA1, SGA2, and HGA in the 10 test cases. It can be observed that HGA achieves the lowest $\bar{e}$ in 9 out of the 10 cases. Only in the 18-leaf node case, SGA2 outperforms SGA1 and HGA in terms of $\bar{e}$. The APRE value averaged over all 10 leaf node settings achieved by SGA1, SGA2 and HGA are 15.79%, 27.52% and 2.13%, respectively. As the number of leaf nodes increases, the APRE $\bar{e}$ values achieved by SGA1 and SGA2 both increase. However, HGA is able to achieve a consistently low APRE value even in the face of increasing number of leaf nodes. Following a one-tail paired Student's t -test, the p -value between SGA1 and HGA is 0.00859 and the p -value between SGA2 and HGA is 0.001403. Both are significantly less than 0.01. Thus, we conclude that the improvements achieved by HGA over SGA1 and SGA2 in terms of APRE are statistically significant at the 99% confidence level.

Regarding the success rate s (Figure 5(b)), HGA has also outperformed SGA1 and SGA2 in the majority of the test cases. The SR s value averaged over all 10 leaf node settings achieved by SGA1, SGA2 and HGA are 44%, 31% and 86%, respectively. As the number of leaf nodes increases, the SR s values achieved by SGA1 and SGA2 both decrease. The superiority of HGA is more pronounced in larger organizations which contain more than 20 leaf nodes. In those cases, SGA1 and SGA2 fail to locate the best known organization instances, whereas HGA achieves high s values of between 90% to 100%. This indicates that HGA uses fewer candidate evaluations to locate the best organization than the conventional GA approaches. Given that candidate evaluations are very computationally expensive in many real-world systems, it is beneficial to use HGA in such circumstances. Following a one-tail paired Student's t -test, the p -value between SGA1 and HGA is 0.002202 and the p -value between SGA2 and HGA is 0.000643. Both are significantly less than 0.01. Thus, we conclude that the improvements achieved by HGA over SGA1 and SGA2 in terms of SR are statistically significant at the 99% confidence level.

The performance graphs of the median runs (i.e. the 5th best runs in our experiment) achieved by SGA1, SGA2 and HGA are shown in Figure 7. Due to the specially designed genetic operators, HGA is able to locate good solutions faster under most of the circumstances. When the number of leaf nodes is larger (especially over 20), HGA generally scores higher fitness values than SGA1 and SGA2 over the same number of candidate evaluations. HGA is also able

TABLE 2
The Fittest Organizational Structures Generated by HGA

Number of Leaf Nodes	Genome Representation of the Fittest Organization	No. of Mediators	No. of Levels	Total No. of Nodes	Fitness Value (u)
12	33233133233	2	3	18	860.39
14	3233132331323	3	3	23	847.62
16	332313323133233	3	3	25	839.20
18	33233133233133233	3	3	27	832.27
20	443434243434243434	1	4	33	821.60
22	33243434143434243434	2	4	37	813.90
24	43434243434143434243434	2	4	42	810.13
26	443434243434143434243434	2	4	44	802.24
28	44343424343414434342443434	2	4	46	795.96
30	4434342443434414434342443434	2	4	48	790.06

to find better organizations within the maximum number of candidate evaluations compared to SGA1 and SGA2. From Figure 7(c-j), it can be observed that HGA converges significantly faster than SGA1 and SGA2.

The genome representations of the organizational structures generated by HGA with the highest fitness value under each leaf node setting are listed in Table 2. The corresponding representation in hierarchical structures of these fittest solutions are shown in Figure 6. It can be observed that aggregator nodes normally are not assigned many sources to reduce the amount of time needed for each aggregator to collect and analyze information from these sources. Most of the fittest organizations contain genome strings 3323, 33233 and 443434. These baseline structures offer high efficiency and form useful building blocks for larger-scale organizations. As the number of databases (i.e. leaf nodes) increases, HGA attempts to distribute the workload by introducing more mediators and increasing the height of the hierarchy.

5.4 Advantages of HGA over ODML

In ODML [24], hard constraints, equivalence classes, parallel search and model abstraction, are used to assist the search process. Rather than going through a decision tree to verify the constraint requirements as ODML does, HGA incorporates the array representation that already ensures the satisfaction of constraints in maximum height of the structure and the number of databases in the system. Parallel search and model abstraction are also intuitively used in HGA.

In ODML, the nodes are treated in three equivalence classes: 1) the mediators, 2) the aggregators, and 3) the databases. The number of organization alternatives is cut down by discarding organizations which are equivalent to an existing one under the symmetry principle. For instance, the organizations that are symmetrical to each other are equivalent in ODML, and only one should be kept as a candidate.

For the 10 test cases of the P2P IR system, despite the truncation of redundant equivalent organizations, the computational complexity of ODML is $O(2.1^N)$, where N is the total number of leaf nodes. In particular, the number of evaluations needed for the 12-database case is 4,304, and that of the 30-database case is 3,788,734,984. In contrast, HGA uses much fewer evaluations (Table 1). This reduces computation burden, as the calculation of utility functions can be very computationally expensive.

It should be noted that HGA is compatible with all the above mentioned search space reduction measures. However, we maintain the equivalent organizations, for they

Fig. 6. The fittest organizational structures generated by HGA.

Fig. 7. The rates of convergence achieved by the comparison approaches.

may contribute to finding a good solution. This compromise results in a larger search space for HGA, whereas in ODM-L, the elimination of redundant equivalent organizations helps narrow down the search space to a large extent. When equivalent organizations frequently appear, ODM-L has an advantage compared to HGA due to the elimination measure. Nevertheless, in the studied system, HGA still manages to evolve the population of organizations at a reasonable pace and saves time for branch pruning at the same time.

5.5 Advantages of HGA over KB-ORG

Different from ODM-L, KB-ORG [28] emphasizes the use of design knowledge in application and coordination levels. With good knowledge, a system can be designed with relatively affordable cost. However, design knowledge largely depends on the level of expertise of the designer which lacks consistency across different individuals. Design knowledge is not guaranteed to be accurate, and it needs to be updated following changes in the environment.

In the IR model, the main difficulty lies in the coordination of nodes (e.g., how many levels of hierarchy is needed). Assume that the designer has successfully located the best organizations for 12, 14, 16 and 18 databases. He may think that a 3-level hierarchy is the best for the 20-database case as well. This will reduce the search space to 58,327 organizations, but it will miss out the highest rated organization, which is 4-leveled with an utility value of 821.60. The utility of the best 3-level organization is 814.11, which is lower than the worst utility (820.01) found by HGA within 50,000 evaluations in all runs. On the other hand, if the designer reaches a relaxed bound of structure height of either 3 or 4 for the 20-database case, the number of organization evaluations will increase to 2,120,662 which is prohibitively large for manual design. Although design knowledge could bring convenience, it is sometimes far from satisfactory. In contrast, HGA is able to handle these test cases without human intervention.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel unique mapping between organizational structures and genome representations and an enhanced genetic algorithm – HGA – to solve the problem of generating proper hierarchical organizational structures in IoT systems. Complementary to existing methodologies that emphasize the pruning of the search space, our algorithm uses a bio-inspired evolutionary approach to lead the search to promising areas of the search space. Thus, it is suitable for generating organizational structures where designer expertise alone is not enough to efficiently produce the appropriate organization instances. In the context of P2P IR systems, we have experimentally shown that the algorithm is able to discover competitive baseline structures in different systems, and assemble them to obtain the highest rated organizational structure. Moreover, the new crossover and mutation operators help HGA enhance its performance in terms of accuracy and stability.

Organization is an important mechanism for communities to improve the efficiency of utilizing their resources. In

businesses, well-defined roles with responsibilities, compensations, and chains of commands are necessary to ensure the successful operation of a company. Even in the emerging field of crowdsourcing which has been envisioned to be a new dimension for enhancing next generation IoT systems [37], [38], human intelligence tasks are moving from independent micro-tasks towards hierarchical tasks that need to be completed by people with different expertise playing different roles [39], [40], [41], [42], [43]. For IoT applications, HGA can serve as a convenient decision support tool to help designers generate various organizational structures quickly, assess their relative merits, and refine the final organizational structures based on these initial designs. In this way, instead of coming up with the organizational structure design for each online community from scratch, organization designers will be able to refine their work from HGA generated templates which provide high quality solutions for the specified requirements. With an automated tool like the HGA, the designers' effort can be directed quickly towards the most promising organizational structures based on the specified system level objectives. This saves the time and effort of organizing large scale systems with massive number of participants, and makes efficient and adaptive mass collaboration possible for future online communities.

With necessary modifications, HGA is applicable to other organizational models as well. It can be used to generate tree-like hierarchical organizations, given that proper fitness functions are provided by designers. Potential application areas include scenario tree and decision tree optimization. On the other hand, the proposed array representation can also be used for other forms of organizations such as holarchies. In our future research, we will explore these directions further.

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative; Nanyang Technological University, Nanyang Assistant Professorship (NAP); the NTU-PKU Joint Research Institute, a collaboration between Nanyang Technological University and Peking University that is sponsored by a donation from the Ng Teng Fong Charitable Foundation; the National Key Research and Development Plan of China (No. 2017YFB1002801); Natural Science Foundation of China under Grant No. 61572471 and No. 61502456; and the Science and Technology Planning Project of Guangdong Province under Grant No. 2015B010105001.

REFERENCES

- [1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.
- [2] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.
- [3] J. M. Watts and N. Purnell, "Singapore is taking the 'smart city' to a whole new level," *The Wall Street Journal*, vol. April, pp. <https://www.wsj.com/articles/singapore-is-taking-the-smart-city-to-a-whole-new-level-1461550026>, 2016.
- [4] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

- [5] D. Helbing and E. Pournaras, "Society: Build digital democracy," *Nature*, vol. 527, no. 7576, pp. 33–34, 2015.
- [6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
- [7] Y. Ding, Y. Jin, L. Ren, and K. Hao, "An intelligent self-organization scheme for the Internet of Things," *IEEE Computational Intelligence Magazine*, vol. 8, no. 3, pp. 41–53, 2013.
- [8] Z. Pan, H. Yu, C. Miao, and C. Leung, "Crowdsensing air quality with camera-enabled mobile devices," in *Proceedings of the 29th AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-17)*, 2017, pp. 4728–4733.
- [9] Y. Zheng, H. Yu, L. Cui, C. Miao, C. Leung, and Q. Yang, "SmartH-S: An AI platform for improving government service provision," in *Proceedings of the 30th AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-18)*, 2018.
- [10] H. Yu, Z. Shen, and C. Leung, "From Internet of Things to Internet of Agents," in *Proceedings of the 2013 IEEE International Conference on Internet of Things (iThings'13)*, 2013, pp. 1054–1057.
- [11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [12] K. M. Carley, "Computational organization science: A new frontier," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 3, pp. 7257–7262, 2002.
- [13] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a social structure to the Internet of Things," *IEEE Communications Letters*, vol. 15, no. 11, pp. 1193–1195, 2011.
- [14] H. Ning and Z. Wang, "Future Internet of Things architecture: Like mankind neural system or social organization framework?" *IEEE Communications Letters*, vol. 15, no. 4, pp. 461–463, 2011.
- [15] A. E. Eiben and J. Smith, "From evolutionary computation to the evolution of things," *Nature*, vol. 521, no. 7553, pp. 476–482, 2015.
- [16] F. Battiston, V. Nicosia, V. Latora, and M. S. Miguel, "Layered social influence promotes multiculturalism in the Axelrod model," *Scientific Reports*, vol. 7, no. 1809, pp. doi:10.1038/s41598-017-02040-4, 2017.
- [17] N. R. Jennings, L. Moreau, D. Nicholson, S. Ramchurn, S. Roberts, T. Rodden, and A. Rogers, "Human-agent collectives," *Communications of the ACM*, vol. 57, no. 12, pp. 80–88, 2014.
- [18] J. A. Stankovic, "Research directions for the Internet of Things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.
- [19] V. Lesser, "Reflections on the Nature of Multi-Agent Coordination and Its Implications for an Agent Architecture," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 1, no. 1, pp. 89–111, 1998.
- [20] L. Pan, X. Meng, Z. Shen, and H. Yu, "A reputation pattern for service oriented computing," in *Proceedings of the 7th International Conference on Information, Communications and Signal Processing (ICICS'09)*, 2009, p. doi:10.1109/ICICS.2009.5397618.
- [21] H. Yu, Y. Cai, Z. Shen, X. Tao, and C. Miao, "Agents as intelligent user interfaces for the net generation," in *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI'10)*, 2010, pp. 429–430.
- [22] Q. Wu, X. Han, H. Yu, Z. Shen, and C. Miao, "The innovative application of learning companions in virtual singapura," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'13)*, 2013, pp. 1171–1172.
- [23] G. Pickard, W. Pan, I. Rahwan, M. Cebrian, R. Crane, A. Madan, and A. Pentland, "Time-critical social mobilization," *Science*, vol. 334, no. 6055, pp. 509–512, 2011.
- [24] B. Horling and V. Lesser, "Using quantitative models to search for appropriate organizational designs," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 16, no. 2, pp. 95–149, 2008.
- [25] J. A. Baum and J. V. Singh, *Evolutionary Dynamics of Organizations*. Oxford, UK: Oxford University Press, 1994.
- [26] M. Wooldridge, N. Jennings, and D. Kinny, "The gaia methodology for agent-oriented analysis and design," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285–312, 2000.
- [27] J. Vázquez-Salceda, V. Dignum, and F. Dignum, "Organizing multiagent systems," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 307–360, 2005.
- [28] M. Sims, D. Corkill, and V. Lesser, "Automated organization design for multi-agent systems," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 16, no. 2, pp. 151–185, 2008.
- [29] B. Li, H. Yu, Z. Shen, and C. Miao, "Evolutionary organizational search," in *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, 2009, pp. 1329–1330.
- [30] L. Yu, Z. Shen, C. Miao, and V. R. Lesser, "Genetic algorithm aided optimization of hierarchical multi-agent system organization," in *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, 2011, pp. 1069–1070.
- [31] B. Li, H. Yu, Z. Shen, L. Cui, and V. R. Lesser, "An evolutionary framework for multi-agent organizations," in *Proceedings of the 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'15)*, 2015, pp. 35–38.
- [32] E. Mezura-Montes, "Evolutionary computation: A unified approach," *Artificial Life*, vol. 13, no. 4, pp. 423–426, 2007.
- [33] G. Nan, M. Li, and J. Kou, "Multi-level genetic algorithm (MLGA) for the construction of clock binary tree," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05)*, 2005, pp. 1441–1445.
- [34] C. Aranha and H. Iba, "The memetic tree-based genetic algorithm and its application to portfolio optimization," *Memetic Computing*, vol. 1, no. 2, pp. 139–151, 2009.
- [35] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of Things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.
- [36] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995, pp. 24–31.
- [37] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, 2011.
- [38] R. Silberzahn and E. L. Uhlmann, "Crowdsourced research: Many hands make tight work," *Nature*, vol. 526, no. 7572, pp. 189–191, 2015.
- [39] S. Nath and B. M. Narayanaswamy, "Productive output in hierarchical crowdsourcing," in *Proceedings of the 13th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'14)*, 2014, pp. 469–476.
- [40] H. Yu, C. Miao, Z. Shen, and C. Leung, "Quality and budget aware task allocation for spatial crowdsourcing," in *Proceedings of the 14th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'15)*, 2015, pp. 1689–1690.
- [41] C. Miao, H. Yu, Z. Shen, and C. Leung, "Balancing quality and budget considerations in mobile crowdsourcing," *Decision Support Systems*, vol. 90, pp. 56–64, 2016.
- [42] H. Yu, C. Miao, C. Leung, Y. Chen, S. Fauvel, V. R. Lesser, and Q. Yang, "Mitigating herding in hierarchical crowdsourcing networks," *Scientific Reports*, vol. 6, no. 4, pp. doi:10.1038/s41598-016-0011-6, 2016.
- [43] H. Yu, C. Miao, Y. Chen, S. Fauvel, X. Li, and V. R. Lesser, "Algorithmic management for improving collective productivity in crowdsourcing," *Scientific Reports*, vol. 7, no. 12541, pp. doi:10.1038/s41598-017-12757-x, 2017.

Zhiqi Shen is a Senior Research Scientist with the School of Computer Science and Engineering (SCSE), Nanyang Technological University, Singapore. He obtained B.Sc. in Computer Science and Technology from Peking University, M.Eng. in Computer Engineering in Beijing University of Technology, and PhD in Nanyang Technological University, respectively. His research interests include Multi-agent Systems (MAS), Goal Oriented Modeling, Agent Augmented Interactive Media, and Interactive Storytelling.

Han Yu is a Nanyang Assistant Professor (NAP) at the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore. He received his B.Eng. (Hons) degree and Ph.D. degree from SCSE, NTU in 2007 and 2014, respectively. He held the prestigious Lee Kuan Yew Post-Doctoral Fellowship (LKY PDF) from 2015 to 2018. His research focuses on stochastic optimization for data-driven algorithmic management in mass collaboration systems.

Victor R. Lesser received his B.A. in Mathematics from Cornell University in 1966, and the Ph.D. degree in Computer Science from Stanford University in 1973. He then was a post-doc/research scientist at Carnegie-Mellon University, working on the Hearsay-II speech understanding system. He has been a professor in the School of Computer Science at the University of Massachusetts Amherst since 1977, and was named Distinguished University Professor of Computer Science in 2009.

His major research focus is on the control and organization of complex AI systems. Professor Lesser is a Founding Fellow of the American Association of Artificial Intelligence (AAAI), and is considered a leading researcher in the areas of blackboard systems, multi-agent/ distributed AI, and real-time AI. He has also made contributions in the areas of computer architecture, signal understanding, diagnostics, plan recognition, and computer supported cooperative work. He has worked in application areas such as sensor networks for vehicle tracking and weather monitoring, speech and sound understanding, information gathering on the internet, peer-to-peer information retrieval, intelligent user interfaces, distributed task allocation and scheduling, and virtual agent enterprises.

Ling Yu worked as a research associate in the School of Computer Engineering, Nanyang Technological University (NTU), Singapore. She received her M.Eng. from the School of Electrical and Electronics Engineering, NTU in 2010, and her B.Sc. in Mathematics and Applied Mathematics from Fudan University in 2007. Her research focuses on evolutionary computing. She is now a Senior Risk Analyst at PayPal Shanghai, China.

Chunyan Miao received the B.S. degree from Shandong University, Jinan, China, in 1988, and the M.S. and Ph.D. degree in Nanyang Technological University, Singapore, in 1998 and 2003, respectively. She is currently a Professor in the School of Computer Science and Engineering at Nanyang Technological University (NTU), and the Director of the Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY). Her research focus is on infusing intelligent agents into interactive new media (virtual, mixed, mobile and pervasive media) to create novel experiences and dimensions in game design, interactive narrative and other real world agent systems.

Her research focus is on infusing intelligent agents into interactive new media (virtual, mixed, mobile and pervasive media) to create novel experiences and dimensions in game design, interactive narrative and other real world agent systems.

Yiqiang Chen received the B.S. and M.S. degrees in computer science from Xiangtan University, Xiangtan, China, in 1996 and 1999, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003. In 2004, he was a Visiting Scholar Researcher with the Department of Computer Science, Hong Kong University of Science and Technology (HKUST), Hong Kong. He is currently a professor and the director of the Pervasive Computing Research Center at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). His research interests include artificial intelligence, pervasive computing, and human computer interaction.

Computing Research Center at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). His research interests include artificial intelligence, pervasive computing, and human computer interaction.