An Evolutionary Framework for Multi-Agent Organizations

Boyang Li*, Han Yu‡, Zhiqi Shen†‡, Lizhen Cui§ and Victor R. Lesser¶
*Georgia Institute of Technology, Atlanta GA, USA
†School of Computer Engineering, Nanyang Technological University (NTU), Singapore
‡Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), NTU, Singapore
§School of Computer Science and Technology, Shandong University, Jinan, China
¶School of Computer Science, University of Massachusetts Amherst, Amherst MA, USA
boyangli@gatech.edu, {han.yu, zqshen}@ntu.edu.sg, clz@sdu.edu.cn, lesser@cs.umass.edu

***Abstract*—The organizational design of a multi-agent system (MAS) is important for its efficiency, adaptability and robustness. However, finding suitable organizational structures for different MASs is a challenging problem. In this paper, we propose a Framework of Evolutionary Optimization for Agent Organizations (FEVOR) based on Genetic Programming for optimizing tree-structured MASs. FEVOR employs a flexible representation of organizations and may be applied to a wide range of organizational forms such as pure hierarchies, holarchies, and federations. Compared to existing work, FEVOR is capable of efficient quantitative search and less vulnerable to stalling at local optima due to its non-greedy nature. Extensive experiments for optimizing an information retrieval system have been conducted to demonstrate the advantages of FEVOR in generating suitable MAS organizations for adaptive environments.**

***Keywords*-Agent organization; evolutionary computing;**

I. Introduction

In today's information technology landscape, many applications can be modelled as multi-agent systems (MASs) in which human beings and autonomous entities dynamically collaborate to achieve their goals [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. The organization of MASs should adapt to its environment and the tasks at hand to meet performance requirements. Therefore, it may be necessary to optimize MAS organizations at both design time and run time. Optimizing the design of MAS organizations has been an active field of research, with many studies devoted to optimization of a single form of organizations and general methodologies as guidelines for human designers [13]. Several quantitative frameworks and methods of organizational optimization have also been proposed, such as KB-ORG [14], and Organizational Design Modeling Language (ODML) [15]. These existing approaches mainly employ either exact search techniques or greedy optimization methods. However, as the optimization complexity has been shown to be *NEXP-complete* [15], it is generally infeasible to apply exact search methods in such optimization problems. On the other hand, we may wish to obtain globally optimal results, which greedy algorithms might fail to provide within nonlinear and non-monotonic search spaces. As a result, a heuristic global search method for optimizing MAS organizations is needed.

In this paper, we extend the EOS approach in [16] to propose a Framework of Evolutionary Optimization for Agent Organizations (FEVOR) based on Genetic Programming. FEVOR optimizes tree-structured organizations. It is well suited for hierarchical organizations as well as a number of other tree-like organizational forms such as holarchies and federations [17]. It provides common types of structures generalized from real applications to simplify the design process while providing the flexibility of allowing the user to customize those types and specify hard constraints. Strong type support is provided to ensure hard type constraints are obeyed by stochastically generated solutions. Most importantly, the evolutionary nature of the algorithm is less prone to stalling at local optima and may find the global optimum, hence filling the gap between complex exact searches and greedy methods that stall at local optima.

II. Related Work

A number of optimization methods based on evolutionary computation for agent organization have also been proposed. However, they generally focus on only one special niche of organizational forms or a particular aspect or process in organizational design. The evolutionary technique proposed in [18] imitates the game of Life [19] as an attempt to optimize partitioning of the information space in an information retrieval (IR) system. In [20], a genetic algorithm employing 2-dimensional genomes is proposed to optimize coalition formation. The mutation and crossover operators are designed to ensure validity of the descendants. Although the search space can generally be reduced by considering only a single process, the risk is that a locally optimal solution of that process may not be part of the globally optimal solution, as multiple facets of the system may be interdependent. Hence, it may be necessary to optimize the entire organizational control simultaneously. The ability of FEVOR to achieve this is attributed, in part, to its tree representation, which can express much more choices in organizational design compared to 2-D representations employed by previous methods.

III. The FEVOR Framework

The FEVOR algorithm is shown in Algorithm 1. A population consisting of a number of possible MAS organizations is first randomly initialized. Each organization is one possible solution to the problem and is called an *individual*. During each iteration of the algorithm, some individuals with high fitness are selected to enter into the next population. Many selection mechanisms are possible, and we adopt the Roulette selection where probability of selecting one individual is proportional to the ratio between its fitness and the sum of fitness of all individuals in the MAS. With a probability, each of the selected individuals may be mutated. The resulted population then replaces the old population. The algorithm optimizes the population as the pressure of stochastically selecting the fittest individual solutions will gradually push the population towards optimal solutions.

We design five types of nodes to represent the organization in the solution tree. They are the *AgentNode*, the *ResourceNode*, the *RoleNode*, the *NullNode* and the *GroupNode*. The five primitive node types provide abstractions from real-world applications. With these building blocks, flexible tree structures can be constructed to represent and optimize a wide range of MAS organizations. However, users of FEVOR should not directly use the primitive types, except the NullNode type. Instead, they should derive their own types from the other four primitive types by specifying a type name. For NonTerminal node types, they must also specify a constraint which indicates the number and types of children allowed for that node type. The user may also supply properties for any derived types. These properties may be used in the evaluation of solution candidates. In the next several paragraphs, we explain the five types of nodes and the use of constraints.

Algorithm 1 The Evolution Algorithm

```
1: Initialize the population $P$ with $n_{pop}$ number of individuals
2: Number of generations $k = 0$
3: while $k < max\_gen$ and population fitness not converged do
4:    Create an empty population $P'$
5:    for $j = 0; j < n_{pop}; j++$ do
6:       Select an individual $i$ in $P$ based on fitness
7:       Select either redistribution mutation (see Algorithm 2) or point mutation
8:       With probability $P_{mutation}$, mutate $i$ accordingly
9:       Put $i$ into $P'$
10:      Evaluate the fitness value of $i$
11:   end for
12: end while
```

1) GroupNode: This non-terminal node is used to represent a group of agents. For example, all agents reporting to the same aggregator and the aggregator itself in the IR system may be considered as such a group. All aggregators under one mediator and the mediator itself constitute another group. The root node of all organizational trees should be a GroupNode. A GroupNode can have children of any type derived from the primitive types.

2) AgentNode: An AgentNode represents an agent that is created with computational resources and may consume task resources. Since an AgentNode may have some ResourceNodes as its children, it is a NonTerminal node. There are also agents performing solely coordination tasks, which do not directly utilize task resources. Those agents have a NullNode as their only child nodes. An AgentNode can only have children of types derived from the ResourceNode or children of the type NullNode.

3) RoleNode: RoleNode is specially designed for cases where an agent may play multiple roles. For example, in each group of controlling agents, there are one verifier, one manager and one handler role that need to be performed by these agents. One RoleNode represents a role, and its value will indicate the agent taking up that role. The use of RoleNode allows one agent to assume multiple roles, so as to balance agent loads and conserve computational resources.

4) ResourceNode: The ResourceNode is another Terminal node. It represents any other resources, other than computational resources, that are needed for fulfilling organizational goals. The number of instances of each type of ResourceNode must be specified before the optimization commences. All ResourceNode must be involved in the organization and they cannot be created dynamically.

5) NullNode: The NullNode is the only node from which no new node types can be derived. Its only function is to fill the leaf position in the tree as the only child of AgentNodes which do not require ResourceNodes.

As mentioned earlier, we take a strongly typed approach of Genetic Programming. Hence, for each NonTerminal node, the designer needs to specify the types of nodes which are allowed to become its children. This is expressed as a constraint for each derived type of NonTerminal nodes.

6) Limit on Resources: The last thing a designer needs to specify in order to use FEVOR is the number of resource instances available in the MAS. We assume that all task resources must be fully utilized but the computational resource.

A. Mutation Operators

In this subsection, we first present two mutation operators specially tailored for the optimization of multi-agent organizations: *redistribution mutation* and *point mutation*. The mutation operators allows not only the optimization of adjusting how task resources are shared among the agents, but also what agents should be created with computational

Algorithm 2 The Redistribution Mutation Algorithm

```
1: Randomly select a type $T$ derived from AgentNode or ResourceNode
2: Randomly select two NonTerminal nodes $N_1$ and $N_2$ (One of them has children of type $T$, and the other node can accommodate children of type $T$)
3: Remove all children of type $T$ from the two nodes and put them in the list $L$
4: if the constraint of the $N_1$ specifies it needs $m$ compulsory children of type $T$ then
5:     Randomly select $m$ nodes from $L$
6:     Remove them from $L$ and add them to children of $N_1$
7: end if
8: Do the same for $N_2$
9: for each node $o$ left in $L$ do
10:     Randomly select $N_1$ or $N_2$ as its parent
11:     Remove $o$ from $L$
12: end for
```

resources to participate in the competition for task resources, and how to organize computational resources in the basic units of agents. In line with our strongly typed approach, the mutation operations generate only valid organization trees so as to reduce the search space. As we consider organizational optimization as a resource allocation problem, it is necessary to fine-tune the allocation of task resources between agents and computational resources between groups of agents, especially when the agents are heterogeneous. This is accomplished through the redistribution mutation.

Algorithm 2 shows the pseudo-code for the redistribution mutation. We assume the constraints in the tree are properly specified such that the number of RoleNodes and AgentNode is proportional to the number of ResourceNodes and every GroupNode has at least two children. For a tree with a maximum of n ResourceNodes, the complexity of redistribution mutation is $O(n^2)$.

IV. Experimental Evaluation

In our experiments, we optimize the organization of a peer-to-peer information retrieval (P2P IR) system. Such systems are widely applied in serving users of P2P content sharing systems. A P2P IR system is made up of a set of nodes connected in a peer-to-peer manner. Each node shares a document collection with others. These nodes work in cooperation to provide IR services to users. Such a system can be modeled as a MAS in which each node is represented by an autonomous agent. In such a fully distributed system, the agents' efforts need to be organized in order to find the results for the queries efficiently.

The performance of the IR system is mainly affected by two factors $response\_recall$ and $response\_time$. $response\_recall \in [0, 1]$ is the ratio between the amount of information the system that can be used to answer a query and the total amount of information in the system. $response\_time$ is the time elapse between the system receiving a query and answering it. It is measured in milliseconds. In our experiments, the system utility is considered to be a weighted average of these two values.

$$utility = response\_recall \times 1000 - \frac{response\_time}{10} \tag{1}$$

Here, we are only interested in several key parameters affecting the performance of the IR system. They are the $query\_rate$, $service_rate$s of mediators and aggregators respectively, $search\_set\_size$ and $query\_set\_size$. The $query\_rate$ and $service\_rate$ denote the frequency of arriving queries from the users and the processing capabilities of the corresponding agents. The $search\_set\_size$ refers to the number of mediators the query is forwarded to when one mediator receives a query. Each mediator then reports back if they have the relevant information to solve the given query. After that, a subset of mediators will be selected to answer the query. The value $query\_set\_size$ refers to the size of the set of selected mediators.

A. Experiment Settings and Results

Our GP population includes 100 organizational trees randomly initialized and the evolution lasts for 50 generations. In every generation, parents are selected with 2-tournament, and redistribution and point mutation are given equal chances. Our fitness function is directly adopted from the ODML model described in [15] with the weights mentioned previously in Eq. (1).

Each mediator and aggregator in the MAS organization manages at least 2 agents and at most 20 agents. Fifteen databases are controlled by the IR system. In order to adhere to the ODML model, we assume each agent controls only one database. In the 6 experiments, we set $query\_rate$ to two different values, 0.001 and 0.004, with the $service\_rate$ of both mediator and aggregator fixed at 0.005. We use the value $max\_search\_set\_size$ to cap the value of $search\_set\_size$, which is set to the smaller value of either one less than the total number of mediators or $max\_search\_set\_size$. The value of $max\_search\_set\_size$ is set to three different values: 1, 4, and ∞. The value of $query\_set\_size$ is set to $\max[1, search\_set\_size - 1]$.

In each of the 6 experiments, the 20 runs generally do not converge to exactly the same organizational structure. Therefore, we direct our attention to one crucial factor of the structure: the number of mediators. This is because mediators occupy the highest level of authority in the organization and have a significant effect on system utility. Our experiment shows that organizations having the same number of mediators achieve similar utility values. Experiment results are summarized in Table I.

Table I
EXPERIMENT RESULTS ($service\_rate = 0.005$)

Environment Variables			Results			
Query Rate	Max Search Set Size (S)	Query Set Size	Number of Mediators in the Organization		Fitness	
			Average	Standard Deviation	Average	Standard Deviation
0.001	1	1	1.00	0.00	654.48	134.27
	4	$S-1$	2.05	1.67	809.29	171.00
	∞	$S-1$	2.65	2.66	874.62	281.81
0.004	1	1	2.00	0.00	492.81	19.53
	4	$S-1$	4.65	0.49	694.80	27.80
	∞	$S-1$	4.90	0.79	699.88	22.34

V. CONCLUSIONS

In this paper, we proposed a novel framework of evolutionary optimization for multi-agent organizations based on genetic programming, which we name FEVOR. FEVOR fills the gap in existing quantitative optimization frameworks for agent organizations by providing a heuristic optimization technique. It is less prone to stalling at local optima than greedy search methods, and incurs less computational cost than exact search methods. Our experiments optimize agent organizations for peer-to-peer information retrieval systems with FEVOR. The solutions generated are well adapted to changing environments. FEVOR can be a useful decision support tool to help generate suitable organization structures for multi-agent organization designers to fine tune.

ACKNOWLEDGMENT

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office.

REFERENCES

[1] H. Yu, Z. Shen, and C. Miao, "A goal-oriented development tool to automate the incorporation of intelligent agents into interactive digital media applications," *ACM Computers in Entertainment (CIE)*, vol. 6, no. 2, 2008.

[2] L. Pan, X. Meng, Z. Shen, and H. Yu, "A reputation pattern for service oriented computing," in *ICICS'09*, 2009.

[3] H. Yu, Y. Cai, Z. Shen, X. Tao, and C. Miao, "Agents as intelligent user interfaces for the net generation," in *IUI'10*, 2010, pp. 429–430.

[4] H. Yu, Z. Shen, C. Miao, and A.-H. Tan, "A simple curious agent to help people be curious," in *AAMAS'11*, 2011, pp. 1159–1160.

[5] Z. Shen, H. Yu, C. Miao, and J. Weng, "Trust-based web-service selection in virtual communities," *Journal for Web Intelligence and Agent Systems (WIAS)*, vol. 9, no. 3, pp. 227–238, 2011.

[6] H. Yu, Z. Shen, C. Miao, and B. An, "Challenges and opportunities for trust management in crowdsourcing," in *WI-IAT'12*, 2012, pp. 486–493.

[7] Q. Wu, X. Han, H. Yu, Z. Shen, and C. Miao, "The innovative application of learning companions in virtual singapura," in *AAMAS'13*, 2013, pp. 1171–1172.

[8] S. Liu, H. Yu, C. Miao, and A. C. Kot, "A fuzzy logic based reputation model against unfair ratings," in *AAMAS'13*, 2013, pp. 821–828.

[9] H. Yu, C. Miao, B. An, C. Leung, and V. R. Lesser, "A reputation management model for resource constrained trustee agents," in *IJCAI'13*, 2013, pp. 418–424.

[10] Y. Cai, Z. Shen, S. Liu, H. Yu, X. Han, J. Ji, M. J. McKeown, C. Leung, and C. Miao, "An agent-based game for the predictive diagnosis of parkinson's disease," in *AAMAS'14*, 2014, pp. 1663–1664.

[11] H. Yu, C. Miao, B. An, Z. Shen, and C. Leung, "Reputation-aware task allocation for human trustees," in *AAMAS'14*, 2014, pp. 357–364.

[12] H. Yu, X. Yu, S. F. Lim, J. Lin, Z. Shen, and C. Miao, "A multi-agent game for studying human decision making," in *AAMAS'14*, 2014, pp. 1661–1662.

[13] S. Abdallah and V. Lesser, "Organization-based cooperative coalition formation," in *WI-IAT'04*, 2004, pp. 162–168.

[14] M. Sims, D. Corkill, and V. Lesser, "Knowledgeable automated organization design for multi-agent systems," Tech. Rep., 2007.

[15] B. Horling, "Quantitative organizational modeling and design for multi-agent systems," Ph.D. Dissertation, 2006.

[16] B. Li, H. Yu, Z. Shen, and C. Miao, "Evolutionary organizational search," in *AAMAS'09*, 2009, pp. 1329–1330.

[17] B. Horling and V. Lesser, "A survey of multi-agent organizational paradigms," *Knowledge Engineering Review*, vol. 19, no. 4, pp. 281–316, 2004.

[18] C. Goldman and J. Rosenschein, "Evolutionary patterns of agent organizations," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 32, no. 1, pp. 135–148, 2002.

[19] M. Gardner, *Wheels, Life, and Other Mathematical Amusements.* W H Freeman and Company, 1983.

[20] J. Yang and Z. Luo, "Coalition formation mechanism in multi-agent systems based on genetic algorithms," *Applied Soft Computing*, vol. 7, no. 2, pp. 561–568, 2007.