Emergent Distributed Problem-solving Technique for Self-configuring Systems

Brian McLaughlan[1] and Henry Hexmoor[2]
[1]Department of Computer and Information Science
University of Arkansas – Fort Smith, Fort Smith, AR, USA
[2]Department of Computer Science
Southern Illinois University – Carbondale, Carbondale, IL, USA

Abstract - *The goal of configuring a massive, complex multi-agent system can be viewed as a distributed search problem in which each agent attempts to choose a correct configuration. This research presents a technique that can simultaneously function as the problem decomposition and solution aggregation components in such a distributed search environment. The method is tested in a number of large multi-agent simulations to demonstrate its feasibility. The agents in the system are shown to achieve optimal or near optimal configurations in significantly less time than agents configuring themselves individually.*

Keywords: *Multi-agent systems, distributed search, emergence*

1 Introduction

Distributed search is an open problem in AI research. At its core, it is a problem involving efficiently decomposing or partitioning a problem into sub-problems and allocating those sub-problems to one or more computational elements. Ideally, these elements would then be able to explore their much more limited search space in parallel. Research in distributed search can be broadly classified as dealing with one or more of the following: finding algorithms for appropriate decomposition of problems [1], distribution of sub-problems to computational elements [2], synthesizing results of sub-problems [3], and coordination between elements [4]. The research in this paper proposes a technique that will simultaneously handle problem decomposition and result synthesis.

In this research, a massive multi-agent system has been given the goal of configuring its constituent elements (i.e., the individual agents) for optimal performance of the tasks assigned to those elements. In this scenario, we make several assumptions:

- The size of the organization is such that micromanagement of individual agents is infeasible.
- The complexity of the organization and its operating environment is such that the desired configuration of the average agent is unknown.
- Individual agents may already possess some method for learning.
- In the course of performing tasks, agents will communicate or deal with each other, but agents are not required to be benevolent. They are allowed to choose an appropriate configuration for themselves without considering the needs of other agents.

2 Approach

To facilitate the speed of the search, a key observation is made regarding the performance of the agents. If all other factors are held constant, an agent that is more successful at completing its tasks will typically be better configured for the task than an agent that is less successful. Therefore, when agents interact, the more successful agent will pass appropriate traits to the less successful agent. The greater the relative success, the more strongly the trait is adopted or the more likely the trait will be adopted.

Formally, agent $\boldsymbol{A_i}$ has a Sending function F_i^S:

$$\boldsymbol{O_j}(T+1) \leftarrow F_i^S(x_i(T), \exists!\, x \in \{C, S\}). \tag{1}$$

This function sends a configuration C or success score S to the observation list of the receiving agent, $\boldsymbol{A_j}$, where it can be processed. Similarly, a Receiving function, F_i^R, is defined:

$$\boldsymbol{O_i}(T+1) \leftarrow F_i^R(F_j^S(x_j(T), \exists!\, x \in \{C, S\})). \tag{2}$$

This function receives a configuration C or success score S from agent $\boldsymbol{A_j}$ and stores it in the observation list where it can be processed. Finally, a Processing Observations function, F_i^P, and an Updating Configuration function, F_i^U, are defined:

$$\boldsymbol{B_i}(T+1) \leftarrow F_i^P(\boldsymbol{O_i}(T), \boldsymbol{B_i}(T)). \tag{3}$$
$$\boldsymbol{C_i}(T+1) \leftarrow F_i^U(\boldsymbol{C_i}(T), \boldsymbol{B_i}(T), S_i). \tag{4}$$

In F_i^P, agent $\boldsymbol{A_i}$ merges its list of observations with its beliefs to create new or updated beliefs. Similarly, F_i^U utilizes agent $\boldsymbol{A_i}$'s current configuration, its success score, and its beliefs, including its beliefs regarding other agents' configurations and their success scores, to determine its new configuration.

In the BDICTL model, agents will only execute their intentions if there is one plan available. When more than one plan is present, noting is done. Some research (CITATION) proposes that this could be modified to have the agent pick a random plan. We propose that the random selection could be more appropriately performed by the viral trait model. To do so, the beliefs about the agent's internal decision-making process can be modeled as traits. To illustrate, let the following variables be defined:

- $\boldsymbol{E_i}$ ::= set of events associated with the execution of a task by agent $\boldsymbol{A_i}$
- $\boldsymbol{E_j}$ ::= set of events associated with the execution of a task by agent $\boldsymbol{A_j}$
- $\boldsymbol{C_i}$ ::= set of configurations of $\boldsymbol{A_i}$
- $\boldsymbol{C_j}$ ::= set of configurations of $\boldsymbol{A_j}$

Then, the BDI formalization of the trait adoption process of agent $\boldsymbol{A_i}$ becomes:

$$\begin{aligned}&\mathrm{BEL}((\textstyle\sum \mathit{succeeded}(\boldsymbol{E_i}) / \sum \mathit{done}(\boldsymbol{E_i})\\ &< \mathrm{BEL}\,(\mathit{succeeded}(\boldsymbol{E_j}) / \mathit{done}(\boldsymbol{E_j})))\\ &\rightarrow \mathit{optional} \lozenge \mathrm{INTEND}(\forall c_j \in \boldsymbol{C_j}, \forall e_j \in \boldsymbol{E_j}\\ &\cap (\mathit{context}(c_j) = \mathit{context}(c_j)), \forall c_i \in \boldsymbol{C_i}, \forall e_i \in \boldsymbol{E_i}\\ &\cap (\mathit{context}(c_i) = \mathit{context}(c_i))\,(\mathit{does}(\mathrm{BEL}(c_j))\\ &\wedge \mathit{does}(\neg \mathrm{BEL}(c_i))))). \end{aligned} \tag{5}$$

This definition assumes the existence of the helper function *context* which returns the contextual component of the variable. In essence, the formalization states that if $\boldsymbol{A_i}$ believes its average success on a particular task is less than what it believes $\boldsymbol{A_j}$'s average success to be, then it has the option of eventually intending all planning beliefs of $\boldsymbol{C_j}$ as long as those beliefs have the same context as events associated with the task in question. It will not take any belief that is not in the same context. Additionally, $\boldsymbol{A_i}$ would intend to remove any existing beliefs that have the same context as the task. This has the effect of adopting the beliefs, including potential action plans, of a perceived superior peer.

3 Experimentation

To investigate the viability of the model, a simulation was created to examine several environments and scenarios in which the agents could operate. A screenshot of this simulation is shown in Figure 1. To get a feel for the ability of the model to handle a variety of peer networks, three different network types were utilized. These network types are representative of the types of networks found in natural and artificial systems. Possible configurations were a grid network with each agent connected to its four neighbors, a scale-free network in which a minority of agents have the majority of connections [5], and a completely random network. Each network was populated with 10,000 agents that then linked to other agents in the prescribed manner. These connections represented communication links between peer agents.

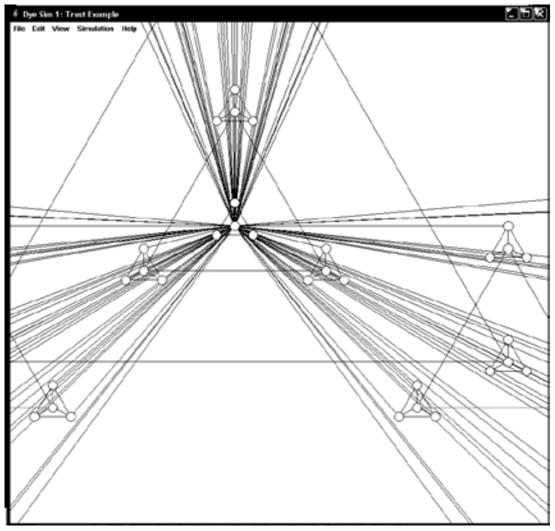

Figure 1: Screenshot of simulation

The multi-agent system was tested in a sales staff scenario. The environment simulated a large department store where the agents acted as the sales staff. The agents were given personality vectors containing five of a possible 100 randomly selected personality characteristics. These five parameters constituted the set of agent Ai's configurations, Ci. In this simulation *personality* was defined as a set of characteristics that uniquely influence cognitions, motivations, and behaviors in various situations [6]. Although there are many different theories on what characteristics constitute one's personality, this research simply assumed the characteristics are those that related to the ability to sale products.

The agents were placed in random departments to await customers. The number of customers was significantly greater than the number of sales agents so that a random "customer drought" would not influence an agent's success rates. Like the sales agents, each customer was given a random personality vector. Customers tended to gravitate towards particular departments based on their personalities and were programmed to be more likely to be persuaded to buy from an agent whose personality was likewise compatible. Therefore, agents with a personality compatible with its assigned department were more likely to be successful than agents with a significantly different personality.

The goal of the system of agents was to have each agent learn and utilize those personality traits that were most appropriate for their department. Each agent was able to perceive the personalities of their peers as the other agents made sales. Essentially, each agent would utilize the Sending function, F^S, to transmit its current personality configuration after each successful transaction.

$$S_i(T+1) \leftarrow F_i^J(S_i(T), 1 - \sum_{x=1}^{5} | C_{cust}(x) - C_i(x) |) \qquad (6)$$

$$S_i(T+1) > S_i(T) \rightarrow \boldsymbol{O_x}(T+1), \forall x \in \boldsymbol{P_i} \leftarrow F_i^S(\boldsymbol{C_i}(T)) \qquad (7)$$

Agent $\boldsymbol{A_i}$'s performance of its job function, F_i^J, is based on the differences between its personality configuration $\boldsymbol{C_i}$ and the customer's configuration C_{cust}. If its score increases, it will send its current configuration to the observation lists of all agents in its peer list, $\boldsymbol{P_i}$.

When adopting the personalities of successful peers, the agent has a chance to assume any of personality traits of the peer. The chance of assuming a trait is based on the difference in the two agents' success rates:

$$Random^x(1{:}100) \leq (S_j - S_i) \rightarrow C_i^x \leftarrow C_j^x, \text{ for x = 1,2,..,5.} \qquad (8)$$

The agents within this simulation were initialized with random values and allowed to transfer their traits to each other. When the system had converged with no more spreading of trait solutions, the simulation stopped, and the results were recorded. Of interest was the system's overall ability to spread the best solutions and the time it took to stabilize.

Next, experiments were performed to test the ability to converge after the insertion of a new agent into the community. A system of agents was allowed to converge, simulating a mature organization. Then, an agent was moved from one region to another. The time to converge and the final fitness of the new organization were recorded.

The ability of this framework was compared to the ability of a duplicate mature organization that utilized individual search to converge. The agents in this organization utilized a greedy binary search method to find appropriate configurations:

1. Try a particular trait for a period of time
2. After the period of time, use binary search, but randomly choose whether to go higher or lower
3. Try this new trait for a period of time
4. After the period of time, evaluate the effectiveness of the solution. If worse than original, try the unchosen direction. If better than original, then continue search. If both higher and lower are worse than original, then finished.

4 Results and conclusions

As shown in Figures 2 and 3, the framework's ability to stabilize at an acceptable state is quite good. The random network had the most variance in convergence time and accuracy, but averaged to reasonable levels. The grid's performance was as expected; when the "best fit" agent was

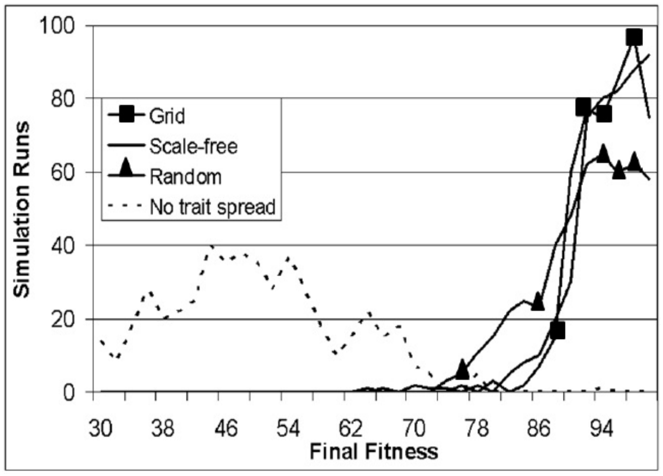

Figure 2: Number of sims obtaining a particular final fitness

centrally located, it took approximately half the distance across the network to stabilize, while "best fit" agents in the corners caused propagation to take longer. The scale-free network had more variation than initially anticipated since the distance across a scale-free network is quite short, but this variation can be attributed to the location of the ideal traits. If the "best fit" agent is also one of the network hubs, the system distributes the trait very rapidly. If the ideal trait is located in a peripheral agent, the transfer of the trait is slow until it hits a major communication artery.

Both the trait-spreading and binary search methods proved successful at quickly acclimating the transported agent to its new environment, as shown in Figure 4. However, the trait-spreading method always succeeded in very few steps, only taking longer when the agent is moved to a border area where nearby successful peers are not actually operating in the same department as the transported agent. This short stabilization time is due to the fact that the agent didn't have to find a successful configuration through trial-and-error, but rather was given a successful starting point from a nearby agent. Additionally, the amount of time take does not depend on the

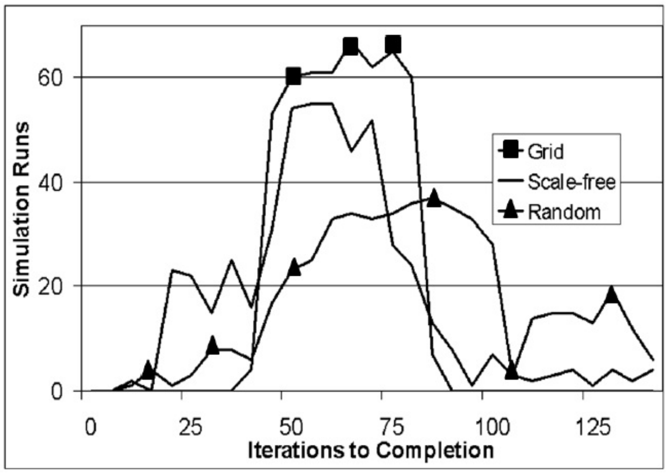

Figure 3: Stabilization times

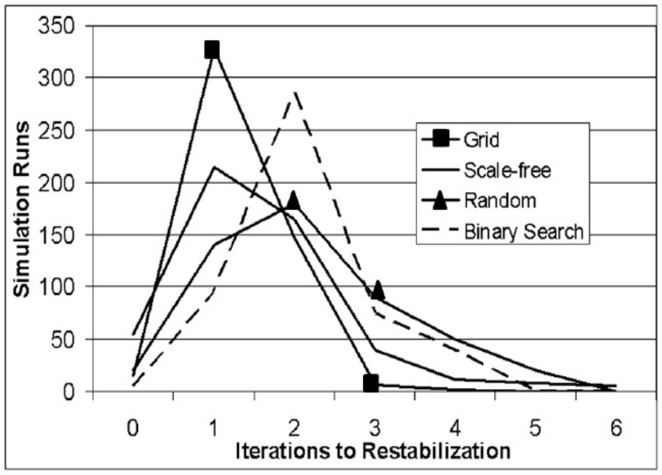

Figure 4: Number of agents with particular stabilization times

size of the problem space, unlike the individual-based binary search.

Although it is not guaranteed to find the ideal solution, it significantly reduces the amount of work done. It naturally produces subproblem boundaries in which similar agents require similar solutions and the aggregates the best known solutions. This work could be combined with other forms of distributed or individualized search. This would serve as the decomposition and aggregation component of such an amalgamation. More research into what types of search would produce the best results is needed. Additional research into the effects of increased parallelization is another area to explore.

5 References

[1] Pynadath, D.V., Tambe, M., Chauvat, N., Cavedon, L. “Toward team-oriented programming.” In *Intelligent Agents VI: Agent Theories, Architectures, and Languages*. 1999.

[2] Armstrong, A., and Durfee, E. “Dynamic Prioritzation of complex agents in distributed constraint satisfaction problems.” In *Proceedings of the 15th International Joint Conferences on Artificial Intelligence (IJCAI)*. 1997.

[3] Yokoo, M., Durfee, E.H., Ishida, T., and Kuwabara, K. “Distributed constraint satisfaction for formalizing distributed problem solving.” In *Proceedings of 12th IEEE International Conference on Distributed Computing Systems*, 1992.

[4] Shen, J., Lesser, V., and Carver, N. “Minimizing communication cost in a distributed Bayesian network using a decentralized MDP.” In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2003.

[5] Amaral, L.A.N, Scala, A., Barthelemy, M., and Stanley, H.E. “Classes of small-world networks” In *Proceedings of the National Academy of Sciences of the United States of America*, 2000.

[6] Ryckman, R.M. *Theories of Personality*. Belmont, CA: Cengage Learning/Wadsworth. 2008.