Efficient Exploration of Environments Using Stochastic Local Search

Ramoni O. Lasisi
Department of Computer and Information Sciences, Virginia Military Institute, Lexington, 24450 VA, U.S.A.
LasisiRO@vmi.edu

Keywords: Agents, Teams, Team Formation, Stochastic Local Search, Simulation, Experiments.

Abstract: This research provides a preliminary investigation of the use of *Stochastic Local Search* (SLS) technique to explore complex environments where agents' knowledge and the time to explore such environments are limited. We model the problem as that of an instance of a *search problem* and develop an SLS technique that enables efficient exploration of such relatively difficult environments by teams of agents. Preliminary results from our experiments using teams of various sizes that implement the model show the effectiveness of the proposed technique. In most cases of the problem instances, teams of agents were able to complete exploration of *more than* 70% of the environments. While in the *best cases*, they were able to complete explorations of *more than 80%* of the environments within short period of time.

1 INTRODUCTION

Autonomous agents in complex environments may need to work together as *teams* to achieve desired goals. This is an important feature of most multiagent environments where individual agents lack all the required capabilities, skills, and knowledge to complete tasks alone. These environments can model real-world problem domains where agents' knowledge and available time to complete tasks in such domains are limited. Agents may thus resort to *team formation* to complete tasks. Team formation or coalition formation are simple models of short-term cooperation (Griffiths and Luck, 2003; Chalkiadakis et al., 2011) where agents complete specific tasks.

Here is a straightforward motivation for the problem we study. Consider a rescue operation in an aircraft crash site where search for survivors may be guaranteed in the first few hours of the crash. Agents neither know where survivors are located nor have enough time to explore the environment for victims. They need, preferably as teams, to devise methods that systematically explore the environment to achieve desired goal. It is not difficult to see that this example and many other similar real-world domains can be modeled as that of *search problems*. This obviously raises the following important question: *How can teams of agents efficiently explore relatively difficult environments using appropriate search strategies that achieve acceptable outcomes?*

This research provides a preliminary investigation of the use of *Stochastic Local Search* (SLS) technique to explore complex environments where agents' knowledge and the time to explore such environments are limited. We model the problem as that of an instance of a *search problem* and develop SLS techniques that enable efficient exploration of such relatively difficult environments by teams of agents.

SLS algorithms have made significant success in solving many hard problems (Hoos and Stutzle, 2005) which involve search of well-defined solutions spaces (or states). A model of SLS algorithms is defined to include a *neighborhood* and an *evaluation function* - both of which are specific to different problems. The goal of an agent using SLS algorithm is to seek a state s from the set of possible states S in the problem domain that optimizes some measures (Neller, 2005). A neighborhood, $N(s)$, is defined for each state s. $N(s)$ is the set of all possible *successor states* that are local to s i.e., the set of all possible states that an agent transits into from the current state s. The evaluation function is defined to exploit the current knowledge of the neighborhood and then *stochastically* selects a successor state $s' \in N(s)$.

This simple method of choosing the successor state by the evaluation function may further be guided towards solutions that optimize goals measures using heuristics. The neighborhood and evaluation function capture two interesting features of SLS algorithms that we exploit in this work. We implement an SLS technique that allows teams of agents to efficiently explore three two-dimensional grid environments.

Lasisi, R.
Efficient Exploration of Environments Using Stochastic Local Search.
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017) - Volume 1*, pages 244-251
ISBN: 978-989-758-219-6

2 RELATED WORK

We discuss SLS in general and provide a description of the three two-dimensional grid environments that we use to evaluate our proposed SLS technique for efficient exploration by teams of agents. Further, we provide some background on team formation and its application in search problems.

SLS algorithms have been successfully applied to many hard problems including Traveling Salesman Problem, Graph Coloring Problem, Satisfiability Problem in Propositional Logic, and more (Hoos and Stutzle, 2000; Hoos and Stutzle, 2005). Common SLS algorithms include *simulated annealing*, *hill climbing*, and evolutionary inspired *genetic algorithms* (Russell and Norvig, 2010). As highlighted in the previous section, the definitions of a neighborhood and its associated evaluation function in SLS algorithms are specific to the problem domain. The real *novelty* in the employment of SLS techniques to construct an algorithm comes from how elegant the neighborhood and the evaluation function are defined for the problem domain such that the algorithm is well-guided towards feasible solutions within a short period of time.

Soule and Heckendorn (Soule and Heckendorn, 2009) describe three environments on which their work is based. We reproduce these environments and there descriptions since we have used them to evaluate our proposed SLS technique. Each of the three environments is composed of two-dimensional grids of 45×45 containing some percentages of *interesting* cells that are distributed in some ways within the environments. A cell is said to be *interesting* if it contains some sub-goals or information that leads to the desired goal of a team. Using the example of the previous section, a cell in this case will be *interesting* if it contains, say, a survivor or victim from the crash.

The environments are named according to how the number of interesting cells are distributed in the grids. They are referred to as *random, clumped,* and *linear*. Figures 1, 2, and 3 depict sample schematic views of random, clumped, and linear environments for a 20×20 two-dimensional grids. The interesting cells in each environments are shown shaded. In a random environment, each cell has a uniform 20% chance of being interesting. For clumped environment, exactly 20% of the cells are interesting and are stochastically clumped in the four corners of the grids. Finally, in the linear environment, exactly 10% of the cells are interesting and they are distributed randomly along eleven rows in the environment.

The same eleven rows are always used, but the exact placement of interesting cells within the rows

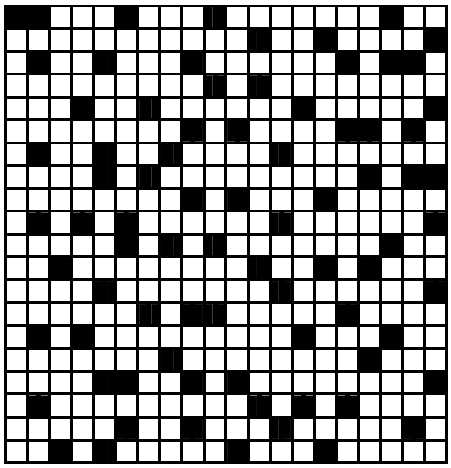

Figure 1: A schematic view of a random environment for a 20×20 two-dimensional grid.

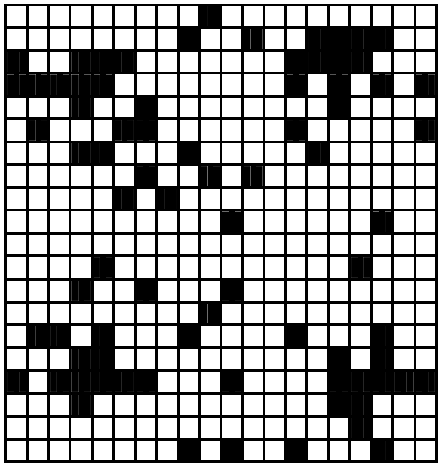

Figure 2: A schematic view of a clumped environment for a 20×20 two-dimensional grid.

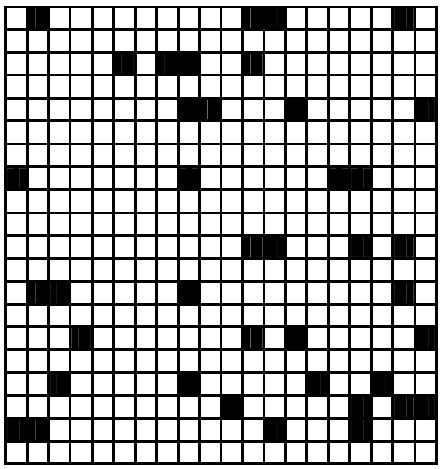

Figure 3: A schematic view of a linear environment for a 20×20 two-dimensional grid.

is random. These environments model applications in the real-world. An environment might represent a minefield with the interesting cells representing positions of potential mines or geological formations (Soule and Heckendorn, 2009). Teams evolved to explore environments may also represent automated planetary surveying team (Thomason et al., 2008).

Soule and Heckendorn use evolutionary algorithms to implement a multiagent team training algorithm called *Orthogonal Evolution of Teams (OET)* to *evolve* teams of agents. The three environments above alternatively serve as both the *training* and *testing* environments to evaluate the performance of their OET algorithm. They consider evolution of *heterogeneous* multiagent teams (i.e., teams of agents with specialized roles). There are two types of specialized agents in their work: *scouts* and *investigators*. The scouts and investigators are respectively responsible for finding as much as possible interesting cells and marking them as investigated. Unlike our approach however, where all agents are limited to moves of *length one* in a single time step in the environments, the scouts are allowed a move of *length two* in a single time step.

Results from our work using SLS technique to explore different environments compare with those of Soule and Heckendorn's with performances within similar ranges. However, it is not yet clear how fair that comparison can be justified since their work employs evolutionary algorithm which come with extensive time requirements of evolutionary learning and huge time and costs of training for agents before they are deployed to actual testing environments.

Team formation is a form of cooperation that has been used in many areas of multiagent systems. Examples can be found in *business* (e.g., organizations form teams to make more sales and hence more profits), in *academia* (e.g., professors form teams to publish articles), in *search and rescue* (e.g., robotic agents form teams in large natural disaster environments to save life and properties), and in *voting* (e.g., voters form teams to win elections). We cooperate with others to solve problems that may be difficult to accomplish alone. This may be due to a number of factors, including how critical a task is, distribution of individual skills/resources, or need for physical presence in multiple work places at the same time.

An interesting number of works have employed various forms of team formation and search strategies in solving problems related to search or exploration. See for example the works of (Batalin and Sukhatme, 2003; Macedo and Cardoso, 2004; Thomason et al., 2008; Hollinger et al., 2009; Ray et al., 2011; Rochlin et al., 2014; Okimoto et al., 2016). It is important to note that none of these research employed the SLS technique that we propose in this work.

3 PROBLEM FORMALIZATION

Given any of the three environments described in the previous section and a number of autonomous agents, each with limited knowledge of the environments, the problem we attempt to solve is to form teams of agents that efficiently explore as much interesting cells as possible in the grids within a limited amount of time. Our attempt in solving this problem uses a model that employs techniques from SLS algorithms.

3.1 The Neighborhood

We present our framework of the *neighborhood* on any of the three environments. Denote by c_{ij} a cell in any grid of an environment where $i, j \in \mathbb{Z}^+$ are the Cartesian coordinates of the cell c_{ij}.

Definition 1. *A state s with a reference cell, c_{ij}, in an environment consists of the reference cell c_{ij}, and all immediate cells $c'_{i'j'}$ from c_{ij} such that $|i-i'| = 1$ or $|j-j'| = 1$.*

An example of a state labeled s is shown in Figure 4 with a reference cell c_{ij}. The immediate cells from c_{ij} are shaded in gray. The set of all possible states S in the problem domain constitutes the search space we seek for feasible solutions (i.e., finding as much as possible interesting cells).

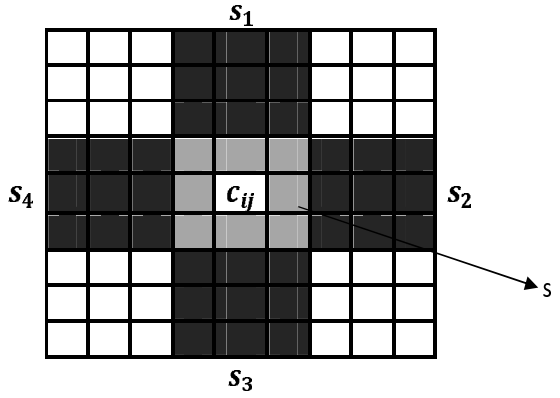

Figure 4: A view of a neighborhood $N(s)$ for a state s. $N(s) = \{s_1, s_2, s_3, s_4\}$.

Definition 2. *The neighborhood $N(s)$ of a state s consists of all states s' that share boundary with s. Hence, any neighborhood consists of only four neighboring states.*

Figure 4 shows an example of a neighborhood $N(s)$ for a state s. States s_1, s_2, s_3, and s_4 (shaded in

black) all share boundary with state s. Thus, $N(s) = \{s_1, s_2, s_3, s_4\}$.

3.2 Evaluation Function

Agents in our model use an *evaluation function* to guide selection of successor states to transit into. Figure 5 gives the pseudocode of the algorithm for our evaluation function.

SuccessorState(a,s)
Input: Agent a and the current state s of a
Output: A successor state $s' \in N(s)$

SuccStates $\leftarrow \varnothing$
foreach state $s' \in N(s)$ **do**
  if s' has not been visited **then**
    if there exists no other agent a' in s' **then**
      SuccStates $\leftarrow$ *SuccStates* $\cup \{s'\}$
if *SuccStates* is empty **then**
    randomly select an s' from $N(s)$
else randomly select an s' from *SuccStates*

return s'

Figure 5: Algorithm SuccessorState(a,s) to implement the evaluation function for exploration of environment using Stochastic Local Search.

Algorithm SuccessorState(a,s) accepts two inputs - an agent a, and the current state s, of the agent. It outputs a successor state s' if one exists, otherwise it stochastically selects any state in $N(s)$ as the successor. In a single time step of exploration of an environment by all agents in our system, each agent calls *SuccessorState*(a,s) algorithm once to determine the next state to transit into. It is not difficult to see that the total running time of a single time step of the exploration of an environment is *linear* in the number of agents, since *SuccessorState*(a,s) algorithm only examines a constant number (i.e., the four) neighboring states to s.

We also show that an agent does not remain infinitely in a particular state in situations where *SuccStates* is empty, at which time the evaluation function stochastically selects any state from $N(s)$. We refer to situation where *SuccStates* is empty as a situation of "no progress". The "no progress" situation is eliminated in the next attempt by the evaluation function to find a successor state and does not occur often as described below.

Suppose an agent a is currently in a state s. Consider a certain attempt t by the evaluation function to find a successor state for a which results in a situation of "no progress". The evaluation function stochastically selects a state s' from $N(s)$ for a to transit into. Now consider the next attempt t' by the evaluation function to find a successor state for a where, as we know, the agent is currently in a new state s' following state s. Observe that the neighborhood for state s' in this attempt t' is different from the neighborhood for state s in the previous attempt t i.e., $N(s') \neq N(s)$ and s is now one of the neighboring states of s' i.e., $s \in N(s')$. Suppose again that attempt t' by the evaluation function to select a successor state for s' results in a situation of "no progress". The evaluation function again stochastically selects a state from $N(s')$. Note that the probability of selecting the state s i.e., the previous state from $N(s')$ as the successor to s' is only $\frac{1}{4}$ as against the probability $\frac{3}{4}$ of selecting any of the remaining three new states from $N(s')$.

Observe that the number of attempts required by the evaluation function until any one of the three states in $N(s')$ apart from state s is selected follows a *geometric random distribution* with probability $p = \frac{3}{4}$. Thus, the *expected* number of attempts required by the evaluation function until any one of these three states is selected is $p\sum_{i=1}^{\infty} i \cdot (1-p)^{i-1} = \frac{1}{p} = \frac{4}{3} < 2$.

3.3 Our Model

We form teams consisting of certain number of agents. One of the team's members is designated as a *leader*. We assume the leader has some additional computational power than other members. The leader is responsible for maintaining an updated status (i.e., visited states) and communicates same to other members when requested. The leader answers the following queries from members: *Has a given state been visited?* and *Is there an agent in a given state?* These are the queries that are used by the evaluation function. Agents are responsible for locating and visiting interesting cells in the grids. All visited cells, either interesting or not are marked as investigated. An agent can move from her current location in only one of four directions (i.e., north, east, west, and south) and is limited to moves of *length one* in a single time step.

When starting all agents (except the leader) are randomly distributed in the environment. We describe the procedure used by agents to explore the environments next. Imagine an agent a currently in a state s that has not been visited. A state is considered visited if the reference cell for the state and all its immediate cells have been marked as investigated. After the exploration of the current state, agent a invokes the evaluation function to determine the successor state to transit into. The successor state guides the decision of the agent on how it *exits* from the reference

cell and the *order* it conducts the search of the current state. Having transits into a successor state, agent a determines if its current cell is interesting, records a score, and marks the cell as visited. The agent then performs an *exhaustive search* of the immediate cells to the reference cell of state s. During the exhaustive search, the agent checks if the cells being searched are interesting, records scores as appropriate, and subsequently marks the cells as visited. On completion of the search of state s, the status of the state (i.e., visited) is communicated to the team leader.

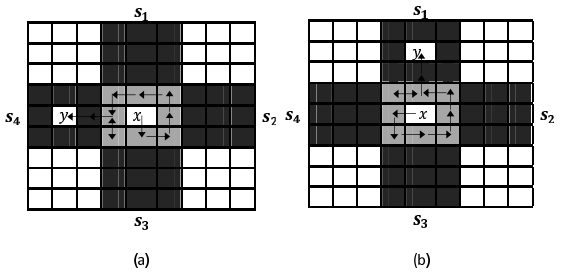

Figure 6: Exhaustive search of a state. (a) Agent exits reference cell x, search current state in the direction of the arrows, and transits into state s_4 with reference y. (b) Agent exits reference cell x, search current state in the direction of the arrows, and transits into state s_1 with reference cell y.

Figure 6(a) provides a simple illustration of an agent currently in a state with reference cell x which later transits into a successor state s_4 with reference cell y. The agent first determines its successor state as s_4 using the evaluation function, then conducts an exhaustive search of the immediate cells to the reference cell x in the direction of the arrows for each time steps, and finally transits into state s_4. A similar example is depicted in Figure 6(b) when the agent transits into state s_1 from the current state. Observe the difference in how the agents in the two figures exit the reference cells of their respective current states and the order in which they conduct their exhaustive search. This difference is due to the fact that the agents transit into different successor states from the current state. At the expiration of the exploration period, we compute the sum of the scores of interesting cells found by each agent as the total score achieved by the team.

3.4 Simulation Results

We present results of our simulations. Figure 7 shows the average percentage of interesting cells found by six-member teams of agents for 45×45 random, clumped, and linear environments using the SLS model for all trials of the experiments. The corresponding standard deviations from the average percentage of interesting cells found by the agent for the three environments are also shown in Table 1.

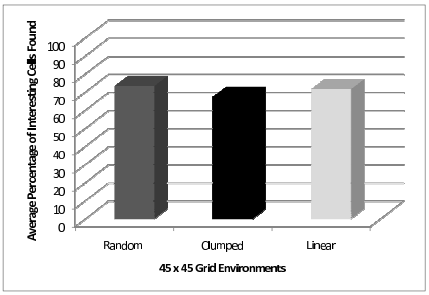

Figure 7: Average percentage of interesting cells found by six-member teams in 45×45 grid environments.

Table 1: Standard deviations from the respective average percentage of interesting cells found by six-member teams in 45×45 grid environments.

	Random	Clumped	Linear
Standard deviation	1.05	1.31	1.09

The average percentage of interesting cells found by agents' teams using the SLS model provides a measure of the level of difficulty of the three environments for the teams. This conversely implies a measure of the relative performance of the teams in each of the environments. Figure 7 shows that the relative performance of the teams in the random environment (~ 74%) is higher than that of the linear environment (~ 72%), which in turn is higher than that of the clumped environment (~ 68%). Thus, the clumped environment appears to be the most difficult of the three environments, followed by the linear, and then, the random environment. The level of difficulty in the three environments may however be assumed to be relatively close considering how small the *spread* (74 – 68 = 6) among the average performance of the teams in the three environment is. This is further evidenced in Table 1 by the tightness of the standard deviations around the average percentage of interesting cells found by agents' teams in the environments.

An implication of the closeness of the level of difficulty of the three environments is that the SLS model's performance has *less reliance* on these environments. Contrarily, (Soule and Heckendorn, 2009) have shown that the performance of the evolved teams by their model depends on both the training and testing environments. They show that training in either the random or clumped environment is a good training for the other environment, but neither is as good of a training environment for the linear environment. In fact, the performance of the evolved teams when they are trained in either of random or clumped, and later deployed in linear environment is poor in comparison with when they are deployed in either of random or clumped environment. Recall also that agents in our model are not subjected to training before being de-

ployed to the testing environments. They only conduct local searches of their environments using two important features from SLS algorithms: neighborhood and evaluation function.

For the second set of experiments, we evaluate the effectiveness of the SLS model by measuring the average percentage of interesting cells found by agents' teams, varying the number of agents in the teams, and the grid sizes in the three environments. Figure 8 shows the average percentage of interesting cells found by agents' teams of different sizes in 45×45 random, clumped, and linear environments. The x-axis indicates the team member sizes while the y-axis is the average percentage of interesting cells found by these teams.

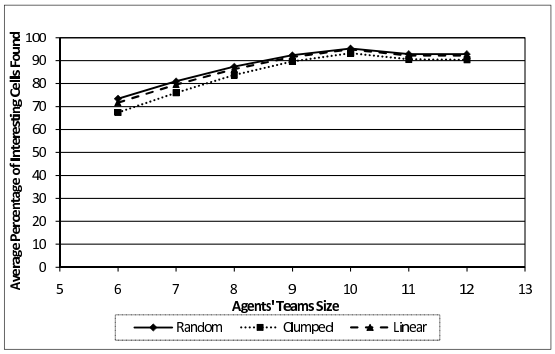

Figure 8: Average percentage of interesting cells found by agents' teams of different sizes in 45×45 grid environments.

The six-member teams always discover more than 70% of the interesting cells for both the random and linear environments, and more than 65% for the clumped environment on the average. As the size of the teams increases, there is a significant increase in the average performance of the teams in the three environments. The average performance of the teams consistently increases with the teams sizes and reaching a peak value of 95% for both the random and linear environments, and 93% for the clumped environment when the team size is ten. It appears that 10-member team is the *optimal* team size when agents implement the SLS model for the three 45×45 grid environments. This can be confirm from Figure 8.

Increasing the number of agents in the teams beyond ten does not appear to improve average performance of agents. We noticed marginal decrease in the average performance of larger teams - as teams' sizes increase past 10, the average percentage of interesting cells found drops below those of the 10-member teams. See Figure 8 for performance of agents' teams of sizes 11 and 12 where the average percentage of interesting cells found by these teams are fairly smaller compared to those of the 10-member teams. Our explanation for this unexpected result is that as the number of agents increases, there is an increase chance of team members revisiting already visited cells. Such efforts by agents does not improve the scores (performance) of the teams since the team has already been rewarded during initial visits of the cells by some members of the team. In other words, some agents in a team may become redundant as the size of the team becomes large.

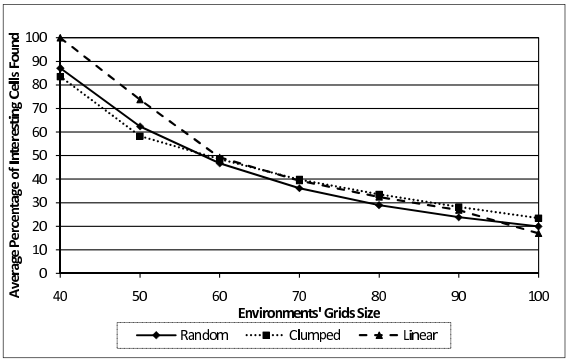

Figure 9: Average percentage of interesting cells found by six-member teams for different grid sizes of environments.

Figure 9 shows the average percentage of interesting cells found by six-member teams for different grid sizes in the three environments. The x-axis indicates the grid sizes while the y-axis is the average percentage of interesting cells found by these teams.

The results show, perhaps not too surprising that in general, the average performance of the teams degrade for the three environments as the dimension of the grids increases. A partial explanation for this is that fixing the team size while increasing the dimension of the environments makes members of the teams to be sparsely distributed in the environments. Thus, it will then be more difficult for agents to cooperate as they now require several time steps to move closer to one another in order to cover different parts of the grids. Nonetheless, even at higher dimensions of the grids, agents' teams are still able to achieve some reasonable level of performance. For instance, when the grid size is 100×100, the 6-member teams found more than 20% of interesting cells for the random and clumped environments but below 20% for the linear environment. Finally, it is important to state that each of the simulations was completed in less than 5 seconds on a Personal Computer with 2.30GHz Pentium (R) Dual-Core Processor and 4GB RAM running Windows.

4 CONCLUSIONS AND FUTURE WORK

We provide a preliminary investigation of the use of *Stochastic Local Search* (SLS) technique to explore complex environments where agents' knowledge and the time to explore such environments are limited. We model the problem as that of an instance of a *search problem* and develop SLS technique that enables efficient exploration of such relatively difficult environments by teams of agents. Thus, we provide initial extensions to (Soule and Heckendorn, 2009)'s work that uses evolutionary algorithms in evolving multi-agent teams in the three environments described in their work.

Experiments using agents' teams of different sizes implementing our model in different problem environments show the effectiveness of our technique. In most cases of the problem instances, teams of agents were able to complete exploration of *more than* 70% of the environments. While in the *best cases*, they were able to complete explorations of *more than 80%* of the environments within short period of time. These results compare with those of Soule and Heckendorn's with performances within similar ranges. However, it is not yet clear how fair that comparison can be justified since their work employs evolutionary algorithm which come with extensive time requirements of evolutionary learning and huge time and costs of training for agents before they are deployed to actual testing environments.

Our model avoids such expensive cost of extensive time requirements of evolutionary learning by agents, the huge time and costs of training agents in particular environments before deployment to actual testing environments. This is made possible as agents in our model are not subjected to training before being deployed to the testing environments. They only conduct local searches of the environments from their current locations using two important features from SLS algorithms: *neighborhood* and *evaluation function*. Further experiments suggest that the level of difficulty of the three environments are relatively the same when agents' teams implement the SLS model. This is evidenced by the closeness of the engendered teams' average performances in the environments. Thus, unlike Soule and Heckendorn's evolutionary model, the SLS model's performance has *less reliance* on the three environments in this work.

There are several areas of ongoing research on this problem. Here are some directions for future work. Future work will consider extension of the SLS model such that average performances of teams are improved across the three environments. We plan to consider how to extend the SLS technique to evolve agents' teams of even larger sizes for environments beyond the 45×45 grid sizes that were the major focus in this work. A drawback of Soule and Heckendorn's model is the unlimited vision of the environments by all agents in their work. We avoid this problem by ensuring that all agents in our model have only limited vision of the environments except the team leader that still has unlimited vision of the environments. We plan to address this issue in future work.

ACKNOWLEDGEMENTS

This research is partially supported by the Virginia Military Institute's Professional Travel Funds.

REFERENCES

Batalin, M. A. and Sukhatme, G. S. (2003). Efficient exploration without localization. In *International Conference on Robotics and Automation*, pages 2714–2719, Taipei, Taiwan.

Chalkiadakis, G., Elkind, E., and Wooldridge, M. (2011). *Computational Aspects of Cooperative Game Theory*. Morgan & Claypool Publishers.

Griffiths, N. and Luck, M. (2003). Coalition formation through motivation and trust. In *International Conference on Autonomous Agents and Multiagent Systems*, Melbourne, Australia.

Hollinger, G., Singh, S., and Kehagias, A. (2009). Efficient, guaranteed search with multi-agent teams. In *Robotics: Science and Systems*.

Hoos, H. H. and Stutzle, T. (2000). Local search algorithms for SAT: an empirical evaluation. *Journal of Automated Reasoning*, 24:421–481.

Hoos, H. H. and Stutzle, T. (2005). *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann.

Macedo, L. and Cardoso, A. (2004). Exploration of unknown environments with motivational agents. In *3rd International Conference on Autonomous Agents and Multi Agent System*, NYC, USA.

Neller, T. W. (2005). Teaching stochastic local search. In *American Association for Artificial Intelligence*.

Okimoto, T., Ribeiro, T., Bouchabou, D., and Inoue, K. (2016). Mission oriented robust multi-team formation and its application to robot rescue simulation. In *25th International Joint Conference on Artificial Intelligence*, New York City, USA.

Ray, D. N., Majumder, S., and Mukhopadhyay, S. (2011). A behavior-based approach for multi-agent q-learning for autonomous exploration. *International Journal of Innovative Technology and Creative Engineering*, 1(7):1–15.

Rochlin, I., Aumann, Y., Sarne, D., and Golosman, L. (2014). Efficiency fairness in team search with self-interested agents. In *13th International Conference on Autonomous Agents and Multiagent Systems*, Paris, France.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Mordern Approach Third Edition*. Prentice Hall.

Soule, T. and Heckendorn, R. B. (2009). Environmental robustness in multiagent teams. In *Genetic and Evolutionary Computation Conference*, Montreal, Quebec, Canada.

Thomason, R., Heckendorn, R. B., and Soule, T. (2008). Training time and team composition robustness in evolved multi-agent systems. *M. O'Neill et al (Eds): EuroGP 2008*, 4971:1–12.