Optimising Search Operations with Swarm Intelligence

Ng Chung Hou*, Lim Wern Han* and Lim Mei Kuan*
*Monash University Malaysia, Selangor, Malaysia.
E-mail: skyler.nch@gmail.com, lim.wern.han@monash.edu, lim.meikuan@monash.edu

***Abstract* — The challenge in search and rescue is to identify the optimal paths when searching the entire location. This is further complicated by the unknown and yet complex environmental terrain; whilst being under the pressure of time. Many of the existing search algorithms such as Depth First Search (DFS) are focused on having only a single agent to sweep through the location. Drawing inspiration from the self-organisation mechanism and the emergence of global behaviour through local interactions between agents in swarm intelligence; this study utilises the information exchange between agents in the swarm to navigate a search area effectively. We demonstrate the proposed swarm-based search method and compare its performance against the existing path finding algorithm Breadth First Search (BFS) on terrains with different complexity. We conducted simulations of search and rescue operations; with findings that the proposed Swarm Intelligence Based Search Strategy (SIS) is able to reach upwards of 95% the effectiveness of BFS with approximately one-fifth the cost of BFS. In addition, a thorough analysis and experimental results to show the optimal number of agents is shown. Our results also demonstrate that having more agents do not necessarily lead to better traversal.**

I. Introduction

This paper aims to identify the optimal number of agents that should be used in a virtual graph or physical terrain search operation. It will be a benchmark study between the proposed search algorithm with the well-known DFS and BFS algorithms. Conceptually, the DFS algorithm deploys a singular agent travelling one node at a time through the path until it reaches its destination. Therefore, DFS may get trapped in parts of the graph that have no goal state and never return. Meanwhile, the BFS algorithm "floods" the graph with agents on all possible paths until it reaches its goal state. When applied to infinite graphs, BFS will eventually find the goal state. Intuitively, having higher number of agents would result in a faster traversal. However, in real scenarios, knowing or deploying the maximum number of agents required for a particular search is not practical. Furthermore, we often have cost (e.g. time & space) associated with search action[1][2].

Intuitively, having higher number of agents would result in a faster traversal. However, in real scenarios, we often have cost associated with search action. Thus, an optimal number of agents and an optimised path traversal mechanism that is cost effective is crucial. In addition, this paper proves that having more agents employed do not necessarily lead to better traversal.

This proposed search strategy draws inspiration from the concept of swarm-based search and traversal methodologies such as particle swarm optimisation[3][4]. Swarm Intelligence is a concept where multiple agents work and communicate with each other to complete certain tasks. The concept is mostly inspired by natural behaviours of animals that works in a swarm like bees or ants, where each of the ant or bee respectively would collectively collaborate to work towards a goal[5]. One particular use for Swarm Intelligence, is search and rescue operations[16][17]. A search problem, both for virtual graphs or physical terrain, would clearly benefit from utilising Swarm Intelligence as a method from traversal and it offers distinctive benefits than when using traditional traversal methods such as DFS or BFS[18].

Search operations can utilise the concept of Swarm Intelligence by introducing multiple agents to the graph, and allowing each agent to collectively communicate information with one another in order to reach the destination in a much shorter time. While at the same time it using an optimal amount of agents to balance the cost of the search traversal, as in using the right amount of agents in traversing the environment instead of assuming that there are an infinite amount of agents[6]. The problem also does not just end with the agents reaching and figuring out the location of its objective, it can be further worked upon to utilise swarm intelligence techniques to optimise the results given[7].

In the field of Multi-Agent Pathfinding (MAPF), there is a sequential variant and the parallel variant. The sequential variants works to minimize the number of steps agents would take to reach the destination, and the parallel variant where it works on reducing the frequency in which agents would move in groups[8]. Generally, there are two main approaches for MAPF. The centralised approach which assumes that all agents in the area are combined into one composite system where the paths would be optimally determined as all knowledge is shared between all agents[9]. The distributed approach which would have each individual agent to

determine its own path and resolve any possible conflicts that may arise[10][19].

The contribution of this paper is two-fold. First, a Swarm Intelligence Based Search Strategy (SIS) is proposed to find the optimal path more effectively. SIS utilises the information exchange between agents in the swarm to navigate a search space effectively. Secondly, we demonstrate that the increase in the number of agents deployed for traversal do not necessarily lead to better performance.

II. Proposed Search Strategy

The process flow of the proposed Swarm Intelligence Based Search Strategy (SIS) is as shown in Figure 1. First is an algorithm to generate random graph, with the ability to set the number of vertex as well as the maximum degree of the vertex. This study will use the modified random walk model to generate randomised graphs. This model of generating random graph is used originally for generating spanning trees[11], however for the purpose of this test, the algorithm has been modified to be able to set the maximum number of edges per vertex. So instead of having one agent "walking" to another vertex to connect both nodes, we have that agent "walk" to multiple vertices. We are using this model to generate the graph because of the simplicity of the algorithm that allowed us to tweak the graphs to accommodate for our configurations for the testing phase, as well as its relatively quick run-time[12].

Second is the algorithm used to traverse the generated terrain. This algorithm is based on the concept of parallel DFS, where the DFS algorithm runs in parallel with each other, cooperating with each other to search the graph with increased efficiency[13][14]. The proposed SIS is able to search the generated graph, where no information of the graph is fed to the algorithm besides the destination node, with a dynamic number of agents that determines how many "instances" of DFS runs simultaneously. This algorithm would have each agent share and exchange information of the paths they have explored so to not have any agents run into a visited vertex, as well as sharing any unexplored area the agents encountered. The pseudocode for the algorithm is as illustrated in Figure 2. The SIS search adopts a top-down approach of traversing the graph, where the number of agents would keep evenly splitting itself into separate groups every time the current vertex it is in has more than one possible route.

An autotester module is developed to automate the simulation for testing. The autotester requests three parameters from the user: the number of nodes for the graph, the maximum degree of each node, and the number of tests each instance has to run.

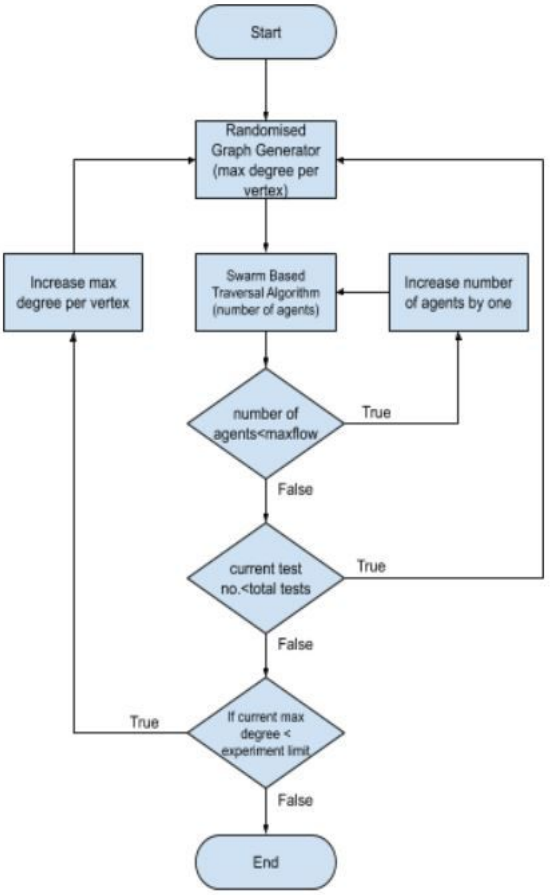

Fig. 1 Flowchart of the experiment

While destination is not reached by any agents:
For each group of agents:
For each neighbour in the current location:
Set current location as visited
If group is in dead end:
Have group travel to next unvisited node
Else:
If only 1 unvisited path:
Move group to the next location
Else If more unvisited paths than agents in group:
Split all agents to all unvisited paths
Remaining paths will remain unvisited
Else more agents in group than unvisited paths:
Evenly distribute agents to groups
Place new groups into the paths

Fig. 2 Pseudocode of the proposed SIS search strategy

With this autotest, each graph generated would be tested by an increasing number of agents traversing it, determined by the max flow of the graph from source to destination. This is because theoretically the maxflow of the graph from source to sink would be the maximum number of agents needed to traverse the graph[15]. Any additional agents added would

not have any further contribution as the graph does not have enough paths for the number of agents traversing, and the surplus agents would be redundant.

The autotest would also reiterate the test for generating the graph from setting the maximum degree of the graph from 4, to the parameter inputted by the user. For each of the iteration, there would be a set number of tests where each test is a different generated graph with the same properties of the current iteration. With this in place it is possible to also test if the maximum degree of the graph would influence the effectiveness of a multi agent graph traversal.

III. Experiment and Result

In the experiment, a total of 69,000 graphs are generated, each with different settings; from 100, 200, and 300 vertices. We iterate the test with a varying maximum degree per vertex from 4 to 26. Each iteration is run over 1000 rounds. In summary, the experiments run over approximately 7,291,000 times (69,000*average max flow). For this paper we will present only the configuration that sets the maximum degree of the graph at 4, 16, and 26.

We compare the performance of the proposed Swarm Intelligence Based Search Strategy with the Breadth First Search algorithm. The BFS is used for benchmarking as the search strategy will always reach completion; or find the goal state. For this study we would consider the sub-optimal solution to be 95%-100% of the shortest path. For example, if the shortest path of the graph takes 100 steps, and the algorithm found a path that takes 105 steps, then the algorithm found a path that is approximately 95.3% (100/105) as efficient compared to Breadth First Search.

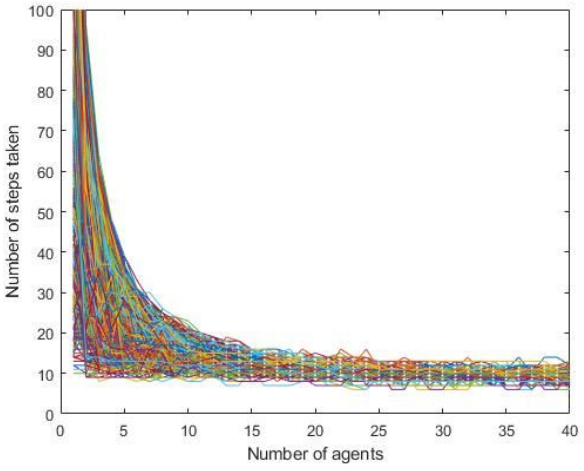

Fig. 3 Size 200 with vertices with maximum degree three, each line within the figure represents one run iteration on the autotest with the mentioned configuration

We discuss our findings in three aspects as follows:

A. Performance of algorithm plateau at approximately 16% of the max flow of the given graph

All of the results returned from the tests, regardless of the configurations, would yield the same pattern as seen in Fig. 2, which is a noticeable performance plateau early on. When we take the suboptimal solution for all iterations with the same configurations and took the average agents it takes to reach our sub-optimal solution, 16% of the maxflow of the graph is the number of agents required for the suboptimal solution. In Fig. 3 we show an example of one of the results with a set configuration, and in Fig. 4 we show the mean of Fig. 3.

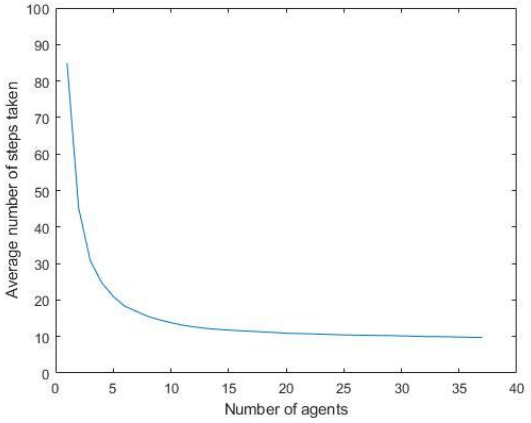

Fig. 4 The mean of all 1000 iterations with the same configuration as Fig. 3

Looking at Table 1 below, we can observe that the number of steps taken decreases as the number of agents involved increases, however the rate of change is not linear as the increase from one to two agents is much more than the increase from ten to forty agents, for all cases.

The performance plateau can be explained by the decreasing likelihood of finding a shorter path than the previous iterations, once a short path is found. Using the well-known DFS algorithm as an example, with the premise that the DFS algorithm will always take a different path than before, to reach the destination. Let's say the first run on the DFS algorithm has returned a path that is 200 steps long, and the shortest available path utilising BFS is 100 steps long. If the next subsequent run, the DFS algorithm returned a shorter path of length 150, then the probability of the next subsequent path to be shorter would be lower. This is because for each possible path there is a standard distribution for the probability that the path would be shorter or longer than the overall average length of all possible paths. So returning to the SIS algorithm, where conceptually it is like running a parallel DFS algorithms, if in the first iteration it returns a path center of the standard distribution, then in the second

iteration will yield an equal probability of returning a shorter or longer path. If it returns a shorter path, then the probability of the subsequent run returning a shorter path would be lowered. Since the proposed SIS algorithm terminates the moment the destination is found, a longer path in subsequent runs will not be returned as the increase in agent count would still yield the same result as the previous iteration, but with the addition of a single agent that might find a shorter path.

Table 1 The average number of steps taken for each agent that is traversing the graph

Size of Graph	Max degree	Number of Agents					
		1 (DFS)	2	5	10	20	40
100	4	41	22	12	9	8	8
100	16	9	6	4	3	3	3
100	26	6	4	3	3	3	3
200	4	84	45	21	13	10	9
200	16	17	10	6	4	3	3
200	26	12	8	5	4	3	3
300	4	130	64	29	18	13	11
300	16	28	15	7	5	4	3
300	26	16	9	5	4	3	3

*It should be noted that in this experiment if only 1 agent is traversing the path, it is equivalent to a depth first search operation. The same logic is applied to Breadth First search as well, if m agents are traversing the path, where m is the maxflow of the graph, then it is equivalent to a BFS operation

Table 2. The efficiency of using agents with 95%-100% effectiveness for graphs of different sizes

Graphs with all vertices with maximum degree n, with different graph sizes	Size of Graph	n=4	n=16	n=26
Average max flow of 1000 graphs generated with vertices with maximum degree n	100	24.6	72.8	76.1
	200	47.6	115	163
	300	71.1	155	228
Average number of agents required to find a path within 5% difference of the actual shortest path (BFS performance)	100	9	13	11
	200	11	19	17
	300	13	20	21
Percentage of maxflow as number of agents needed to achieve >95% efficiency	100	**35%**	**17%**	**13%**
	200	**23%**	**15%**	**10%**
	300	**18%**	**12%**	**9%**

B. The increase in the maximum degree of the graph would improve the results

Building on the previous argument where more agents increases the probability of finding a shorter paths during its run, increasing the maximum degree of the graph would also increase the probability of getting the shortest path, as each vertex would on average contain more possible paths to traverse, allowing for agents to search for more alternative paths in the meantime instead of having agents simply following a group to the next vertex until an alternative path shows up.

C. In the context of the efficacy of the algorithm, the size of the graph does not proportionally scale to its max flow

A larger size of the graph does on average increase the amount of steps taken to reach the destination, however the increase is not proportional to the max flow of the graph. As illustrated in Table 2, the percentage of max flow needed to reach above 95% effectiveness would decrease as the size of the graph increases. This is because as discussed in the first observation, the probability increases and decreases on the iterations, and does not rely on the maxflow of the graph. Therefore, the average number of agents required to achieve above 95% efficiency increases as the size of the graph increases.

IV. CONCLUSIONS

The proposed Swarm Intelligence Based Search Strategy is able to search through the graph efficiently at a highly reduced cost when compared to Breadth First Search algorithm. The statistical results from our experiments show that the SIS search strategy is able to achieve 95% to 100% of the performance of BFS, while requiring only on average 16% of the cost or number of agents required by BFS. Our results also demonstrate that an increase in the number of agents, do not necessarily lead to better performance in swarm-based strategies. We demonstrate that the increase in the number of agents for traversal would reach a plateau. Interestingly, the different terrains do not have significant influence on the optimal number of agents required for search too. This alludes to the information exchange mechanism in swarm intelligence that leads to convergence or emergence behaviour, which is the optimal path in the context of this study.

There are a number of possible directions where this study could explore further. Firstly, is to attempt at a mathematical proof of the probability of achieving 95 - 100% effectiveness with only 16% of the cost. We would also like to investigate if including swarm intelligence methods including Ant Colony Optimisation and Slime Mold Optimisation can be adapted for optimal path finding.

References

[1] Everitt, T., & Hutter, M. (2015, November). Analytical results on the BFS vs. DFS algorithm selection problem. Part I: tree search. In Australasian Joint Conference on Artificial Intelligence (pp. 157-165). Springer, Cham.

[2] Cheung, T. Y. (1983). Graph traversal techniques and the maximum flow problem in distributed computation. IEEE Transactions on Software Engineering, (4), 504-512.

[3] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization (PSO). In Proc. IEEE International Conference on Neural Networks, Perth, Australia (pp. 1942-1948).

[4] Pugh, J., & Martinoli, A. (2007, April). Inspiring and modeling multi-robot search with particle swarm optimization. In 2007 IEEE Swarm Intelligence Symposium (pp. 332-339). IEEE.

[5] Kennedy, J. (2006). Swarm intelligence. In Handbook of nature-inspired and innovative computing (pp. 187-219). Springer, Boston, MA.

[6] Bakhshipour, M., Ghadi, M. J., & Namdari, F. (2017). Swarm robotics search & rescue: A novel artificial intelligence-inspired optimization approach. Applied Soft Computing, 57, 708-726.

[7] Blum, C., & Li, X. (2008). Swarm intelligence in optimization. In Swarm intelligence (pp. 43-85). Springer, Berlin, Heidelberg.

[8] Röger, G., & Helmert, M. (2012, July). Non-optimal multi-agent pathfinding is solved (since 1984). In Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence.

[9] Sharon, G., Stern, R., Felner, A., & Sturtevant, N. R. (2015). Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence, 219, 40-66.

[10] Bennewitz, Maren, Wolfram Burgard, and Sebastian Thrun. "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots." Robotics and autonomous systems 41.2-3 (2002): 89-99.

[11] Lovász, L. (1993). Random walks on graphs: A survey. Combinatorics, Paul erdos is eighty, 2(1), 1-46.

[12] Wilson, D. B. (1996, May). Generating random spanning trees more quickly than the cover time. In STOC (Vol. 96, pp. 296-303).

[13] Rao, V. N., & Kumar, V. (1987). Parallel depth first search. Part I. implementation. International Journal of Parallel Programming, 16(6), 479-499.

[14] Kumar, V., & Rao, V. N. (1987). Parallel depth first search. Part II. analysis. International Journal of Parallel Programming, 16(6), 501-519.

[15] Goldberg, A. V., Hed, S., Kaplan, H., Tarjan, R. E., & Werneck, R. F. (2011, September). Maximum flows by incremental breadth-first search. In European Symposium on Algorithms (pp. 457-468). Springer, Berlin, Heidelberg.

[16] Couceiro, M. S. (2017). An overview of swarm robotics for search and rescue applications. In Artificial Intelligence: Concepts, Methodologies, Tools, and Applications (pp. 1522-1561). IGI Global.

[17] Arnold, R. D., Yamaguchi, H., & Tanaka, T. (2018). Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. Journal of International Humanitarian Action, 3(1), 18.

[18] Khaldi, B., & Cherif, F. (2015). An overview of swarm robotics: Swarm intelligence applied to multi-robotics. International Journal of Computer Applications, 126(2).

[19] Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., ... & Boyarski, E. (2019). Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. arXiv preprint arXiv:1906.08291.