Optimal Fungal Space Searching Algorithms

E. Asenova*, H-Y Lin*, E. Fu, D. V. Nicolau Jr., D.V. Nicolau

***Abstract*— Previous experiments have shown that fungi use an efficient natural algorithm for searching the space available for their growth in micro-confined networks, e.g., mazes. This natural 'master' algorithm, which comprises two 'slave' sub-algorithms, i.e., collision-induced branching and directional memory, has been shown to be more efficient than alternatives, with one, or the other, or both sub-algorithms turned off. In contrast, the present contribution compares the performance of the fungal natural algorithm against several standard artificial homologues. It was found that the space-searching fungal algorithm consistently outperforms uninformed algorithms, such as Depth-First-Search (DFS). Furthermore, while the natural algorithm is inferior to informed ones, such as A*, this under-performance does not importantly increase with the increase of the size of the maze. These findings suggest that a systematic effort of harvesting the natural space searching algorithms used by microorganisms is warranted and possibly overdue. These natural algorithms, if efficient, can be reverse-engineered for graph and tree search strategies.**

***Index Terms*— Maze searching, natural algorithms, biomimetics, microfluidics**

I. Introduction

Biological entities have evolved highly efficient strategies for space searching, which are essential to their survival[1], both as individuals and as species. In many instances, non-human biological algorithms appear to be superior to those used by humans,[2] which opens up the opportunity for their 'reverse engineering' for practical applications.[3] Despite their recent history, the *bio-inspired algorithms* have already achieved a high level of sophistication, allowing for comprehensive surveys and elaborate taxonomy.[3] More than 2000 contributions and more than 40 comprehensive reviews cite "bio-inspired algorithms" in their title. The bio-inspired algorithms, of which many deal with *space-searching* and/or *space-partitioning*, are "inspired" by evolutionary, ecological and swarming processes. Of the possible applications, one could mention many related to space searching and the management of available space[3]: optimal routing for vehicles, e.g., mobile robots, driverless cars, including sensor-based path planning and scheduling; Travel Salesman Problem (TSP), optimal power systems, including distribution over networks, engineering design, e.g., inverse airfoil design. Presently, however, the bio-inspired algorithms are just "inspired" by, rather than "reverse-engineered" from, natural algorithms "developed" by biological entities.

Previous experiments[4, 5] demonstrated that fungi behave very differently in micro-confined spaces, and that they use specific procedures for searching space available for growth. While different species present different variants of this fungal algorithm, its framework is common and it consists of the synergetic use of two distinct 'sub-algorithms': collision-induced branching, and directional memory. These studies also demonstrated that the natural algorithm comprising the two 'sub-algorithms' is markedly superior to variants where one of these is, or both are suppressed.

The present contribution extends the analysis of the efficiency of the fungal space searching algorithm, by benchmarking it against classical space searching algorithms.

II. Methods

A. Fungal space searching algorithms

Previous experimental and stochastic modelling works examined the growth of two species of filamentous fungi, i.e., *Pycnoporus cinnabarinus*,[4] and *Neurospora crassa*,[5, 6] inside a confined maze-like microfluidics structure.

Microfluidics for Harvesting Space Search Algorithms. The fabrication of the test microfluidics structures[4, 6] consisted of the fabrication of a silicon mold; making a negative relief poly(dimethylsiloxane) (PDMS) stamp; rendering the hydrophobic PDMS surface hydrophilic by plasma treatment; and sealing the PDMS onto a flat base glass layer. The enclosed structure had lateral openings, allowing the introduction of inoculation and media. The fungi grow on the basal substrate and explore the test microfluidics structures. All species studied so far present two distinct 'slave' sub-algorithms: *collision-induced branching* and *directional memory*.

Collision-induced branching. This sub-algorithm consists in the growth of the fungus, in confined spaces, without branching, until it reaches a wall, or a corner. If the 'angle of

This paper was submitted for review on 17.2.2016. The work was financially supported by the European Union Seventh Framework Programme (FP7/2007-2013, grant agreements 228971, MONAD; 613044, ABACUS) and Defense Advanced Research Projects Agency, grant agreement N66001-03-1-8913.

E. Asenova, H-Y Lin, and E. Fu are at McGill University, Department of Bioengineering, Montreal, Quebec, H3A 0C3, Canada.

*E. Asenova, H-Y Lin contributed equally to this work

D.V. Nicolau Jr. is at Molecular Sense Ltd., Wallasey, CH44 1AJ, UK

D. V. Nicolau is at McGill University, Department of Bioengineering, Faculty of Engineering, Montreal, Quebec, H3A 0C3, Canada (email: dan.nicolau@mcgill.ca)

attack' is shallow, the hypha, i.e., the filamentous extension by which fungi grow, will slide along the wall. Conversely, if the tip of the hypha, reaches a geometry that does not allow an exit, e.g., a corner, then the hypha will branch.

The fungus has two branching mechanisms: one with very low frequency of branching, which is equivalent to the one used when growing in open spaces; and a high frequency one, triggered by the collision with growth-blocking geometries, which occurs often in confined spaces, e.g., mazes.

Directional memory. The second sub-algorithm consists in the growth process where each branch 'remembers' its initial direction of growth. While each hypha has to negotiate various geometries, whenever the branch has the opportunity to grow in the direction it had initially, it will follow this with a high probability. Moreover, the branch will not grow further if the available space opens in a direction which is more than orthogonal with its initial direction, i.e., angles larger than 90° are not allowed. If no alternative passage is available, the hypha will stop growing and in most instances will branch following a 'rest period' of time.

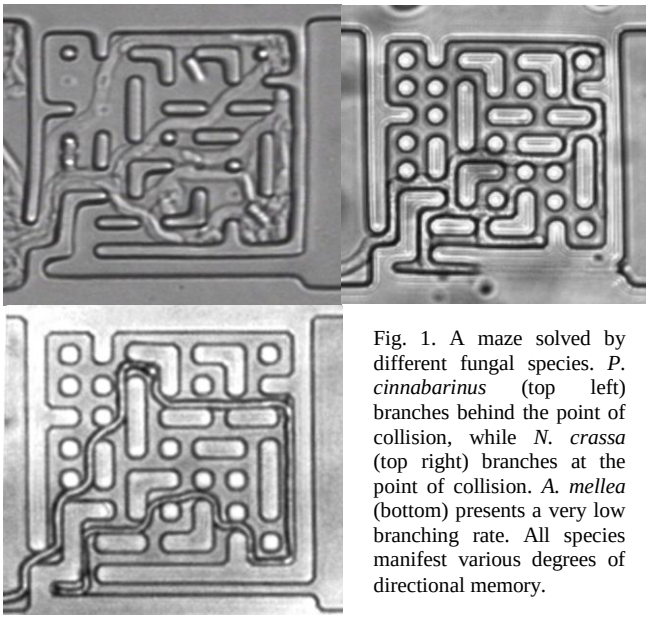

Fig. 1. A maze solved by different fungal species. *P. cinnabarinus* (top left) branches behind the point of collision, while *N. crassa* (top right) branches at the point of collision. *A. mellea* (bottom) presents a very low branching rate. All species manifest various degrees of directional memory.

Different species present different "parametrization" of their space searching algorithm (Fig 1). For instance, upon collision, *P. cinnabarinus* (Fig. 1, top left) will branch away from the leading tip.[4] The hyphae of *N. crassa* (Fig. 1, top right) however will split where the apex touches the collision-inducing geometry.[5, 6] Finally, *A. mellea* (Fig. 1, bottom) presents a very low branching frequency.

In opposition with the species-specific collision-induced branching, the directional memory, which appears to be more important for the efficiency of the fungal space searching algorithm[4], presents smaller species-specific variations. Indeed, Fig. 1 shows that initial hypha entering the maze (left bottom), for all species, maintains its initial direction at least for half of the diagonal of the maze, and that the direction of the growth changes once a new branch has been created.

These two sub-algorithms is presented in Fig. 2, and a sequence of events during fungal growth is presented in Fig. 3.

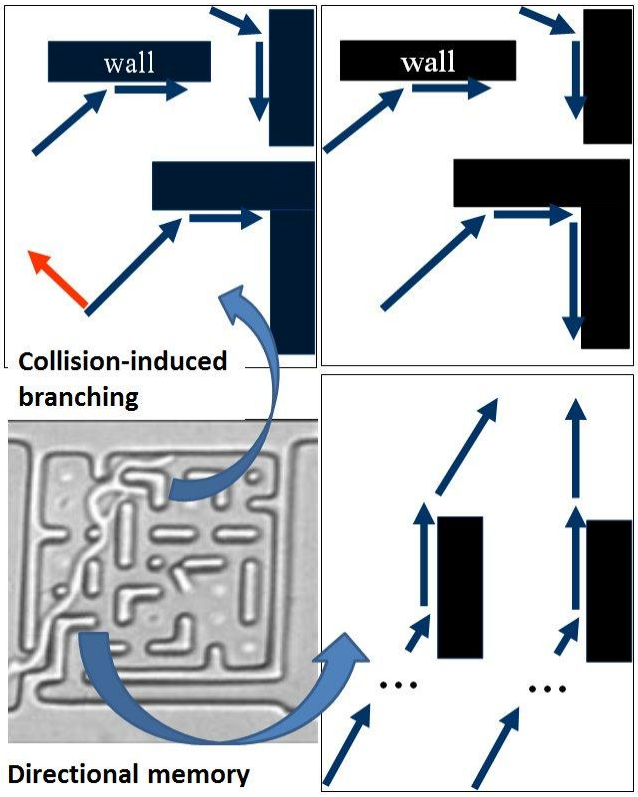

Fig 2. Space searching algorithms used by a fungus (i.e. *P. cinnabarinus*). Top left panel: Collision-induced branching. The hyphae slide along walls if the angle of attack is shallow. When facing a corner, the hyphae will branch (red arrow), unlike "no collision-induced branching" (top right panel). Bottom right panel: Directional memory. The hyphae slide along the walls, then redirect growth in the same direction they had initially (left scheme), unlike "no-directional memory" (right). An image of the fungus growing in a maze (bottom left panel) shows both the collision-induced branching events (top arrow), and the overall directional memory (bottom arrow).

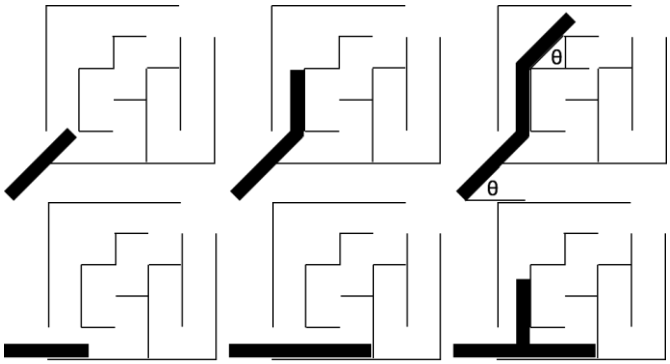

Fig 3. Schematic sequence of fungal growth. Directional memory (top) biases the growth towards the same initial direction (θ). Collision induced branching (bottom) imposes a branching event (here for *P. cinnabarinus*, i.e., branch emerging behind the leading tip) whenever growth is blocked by a dead-end.

B. *Bio-inspired Search Algorithms*

Maze generation. Since maze-solving can be generalized to graph searching, the mazes were generated by a graph-based algorithm (Fig. 4). Specifically, the graph was designed as a "w-by-h" grid, with w and h being the maze width and height. Each cell represents a square on the grid whereas the black lines indicate the walls (Fig 4, left). If each cell is the vertex of a

graph, then a cell may be connected to another cell if there are no walls between them (Fig 4, middle). Each cell can have from 0 to 4 walls; 0 walls means the cell is a 4-way intersection, and 4 walls means the cell is completely closed off.

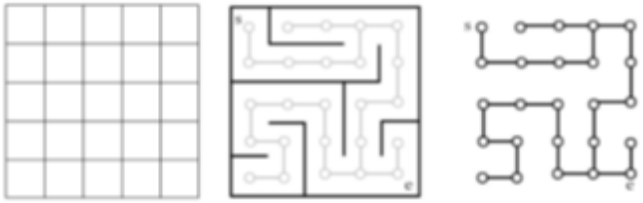

Fig 4. Evolution of the construction of a maze.

The maze is initialized as a 2D array of disconnected vertices, represented as a rectangular grid. Two randomized options are available for maze generation, with distinct algorithms for removing walls in the grid, resulting in two "styles" of mazes. The first algorithm implements a queue-based, Depth-First-Search (DFS) methodology,[7] i.e., a first-in-first-out graph-searching algorithm. Starting from the entrance, a random wall leading to an adjacent cell is removed, and the cell is marked as "visited". The algorithm then removes a random wall in the newly-visited adjacent cell, etc., until the user-supplied exit point is found and all cells have been visited, resulting in mazes with long corridors and few branches.

The second option implements Kruskal's algorithm, [8] the initial maze of closed cells is seen as a grid of disjoint sets, each containing only one vertex. A separate array containing all walls is also created. Then, at random, walls are chosen out of this array. If the wall divides cells belonging to separate sets, the wall is removed and the two cells are joined into one set. This is repeated until all cells belong in a single set. In contrast to DFS, Kruskal's algorithm results in a maze that branches frequently in all directions, and contains few long paths.

Space searching algorithm. The Bio-Inspired Algorithm (BIA) was implemented in Java, coupling two distinct sub-algorithms, i.e., collision-induced branching, and directional memory (pseudo-code in the Appendix A).

Collision-induced branching. Instead of using recursion or a stack, cells were stored in a data structure that allows direct access, in this case, an array. This way, when a dead end is reached, the next cell to visit is determined algorithmically rather than using a last-in-first-out method.

Directional memory. Directional memory is implemented in Java by including the starting angle of each branch object as a class constant, and the current angle of the branch as a class variable. A fungal branch will travel at its starting angle as far as possible, whenever possible, and only changes its current angle when no paths are available in its starting direction.

III. RESULTS AND DISCUSSION

A. Space searching and solving mazes

Mazes are used to estimate the behavioral response of many organisms, i.e., ants, bees, mice, rats, octopi, and humans[9], as well as artificial intelligence-enabled robots.[10]

The efficiency of space searching algorithms depends greatly on the geometry of the space and specifically confinement properties.[11] At one end of the scale, empty space without obstacles cannot be explored any better than by using a diffusion, or diffusion-like approach, e.g. a Levy flight.[12] Depending on how the nutrients (or other resources of interest) are distributed in such a space, it appears generally that Levy flight processes are both what biological systems use and what is actually mathematically optimal.[13]

At the other end of the scale, the space search problem in a maze, a highly constrained geometry, reduces specifically to the problem of graph connectedness.[14] Because the maze is a graph, and it is required for an exit to be found, this translates into asking a computational system, e.g., a fungus, to find if the entry and exit of the maze/graph are connected (and if so, how), or not. This problem, of graph connectedness, is known to be in computational class P and can be solved in a number of ways, but most commonly this is done using "breadth-first search", first proposed in the 1950s.[15] Its time complexity is O(V+E) where V and E are the number of edges and vertices, respectively. In general we think of graphs (and mazes) as being specified by the number of vertices and in the case of a graph were most vertices are locally connected together, the time to establish the path from entry to exit would be a fixed, low (e.g., 1-3) power of E.

B. Fungal 'intelligence'

Microfluidics technology allowed the miniaturization of maze structures, used to test the maze-solving ability of both abiotic[16] and biotic[17] agents and to modulate and observe the collective behavior of bacteria[18, 19].

An interesting aspect of the fungal search algorithms is that they do not require nutrient-related clues regarding the geometry of the environment. Previous studies have documented maze-solving by placement of nutrients at the exits,[17] or quorum-related signaling,[20] but the study of fungal space searching suppressed nutrient gradients. This response is consistent with the fact that natural fungal habitats are nutritionally heterogeneous and require hyphae to continue colony extension in the absence of chemotactic cues.

The ubiquity of fungi in micro-confined maze-like habitats suggests that they may be efficient solving agents of geometrical problems. While this ability was assessed versus variants missing one, or the other sub-algorithm, or both, the performance of the fungal space search algorithm versus standard path search algorithms was not previously examined.

C. Assessing the efficiency of the Bio-Inspired Algorithm

To examine the Bio-Inspired Algorithm (BIA), we tested its completeness, reachable state space and optimality under both non-randomized and randomized mazes, generated by DFS (which resulted in almost no dead ends), and by Kruskal's algorithm (leading to multiple dead ends), respectively.

Completeness. Starting from a root node (beginning of maze), this test aims to find to what extent BIA finds the leaf node (end of maze). The tests were run on different maze sizes and by placing the starting and ending vertices at various

positions. For the maze sizes up to 50x50, BIA solved every maze with different starting and ending positions. Upon encountering a dead end, the algorithm goes back to a previously branching point and resumes from there. While this approach appears to be sub-optimal, the algorithm will always find the exit on finite mazes. In this regard, it must be noted that in many biological instances, robustness of behavior is more important than efficiency.

Reachable state space. Mazes can serve as a background to state-space searching because this is composed of an environment (the maze) that is divided into equally sized units (states). For this test, we defined a start state (beginning of the maze) and final state (end of maze) and counted the number of covered nodes when the final state was reached. In Fig. 5, it can be seen that when the maze size grows, the portion of reachable state space remains constant (around 2/3 is explored).

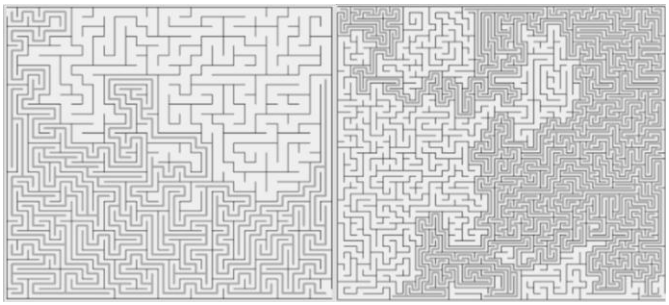

Fig 5. Examples of the coverage of the mazes when explored by BIA, for 20x20 (left), and 50x50 (right) mazes.

Optimality. The objective of this test is to determine the effectiveness of the algorithm in finding a least-cost solution. Since the exit of the maze is unknown, we expect an average of 50% chance of finding the shortest path. As shown by Fig. 6, the reliability of finding the shortest path starts from ~50% in small maze sizes (11x11) and decreases with larger maze sizes. It is important to note that real fungi, tested under laboratory conditions, can pick up various indications in the solution as to where the target is and find it faster. For a more effective search, it can be useful to model those condition, if provided the necessary information.

D. Comparison with other maze solving algorithms

Noting that fungi have complex, and different from computer algorithms, 'objective function' against which they optimize their behavior, it is of critical importance for "mathematical biomimetics" to benchmark these natural algorithms against standard ones with similar scope.

Reachable state space. This test was applied to examine the amount of necessary memory. Because this is an uninformed search (the search is the same regardless of the context), we observed that BIA is less efficient than and an informed search, e.g., A*.[21] However, BIA is consistently better than DFS, i.e. the covered area while the final state was reached for BIA will be larger than informed search but smaller than DFS. While the comparison with inherently more efficient algorithms may be 'unfair', it is nevertheless more comprehensive.

As shown by Fig. 7, DFS takes the most space when performing maze search, while BIA is ~20% more efficient in larger mazes (30x30 and up). A*, being an informed search, is, as expected, much more efficient and therefore much more compact.

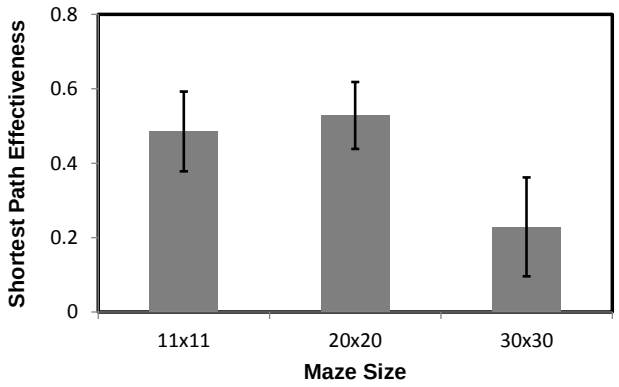

Fig 6. Shortest path effectiveness, as ratio of the length of the shortest path versus the lengths of actual paths, in different maze sizes.

It is important to note the difference between randomized and nonrandomized mazes. In the non-randomized maze, BIA and DFS are more similar in performance due to the limited number of dead ends (Fig 7, top). With randomized mazes where they are multiple dead ends, the difference between BIA and DFS is visible even with smaller mazes (Fig 7, bottom).

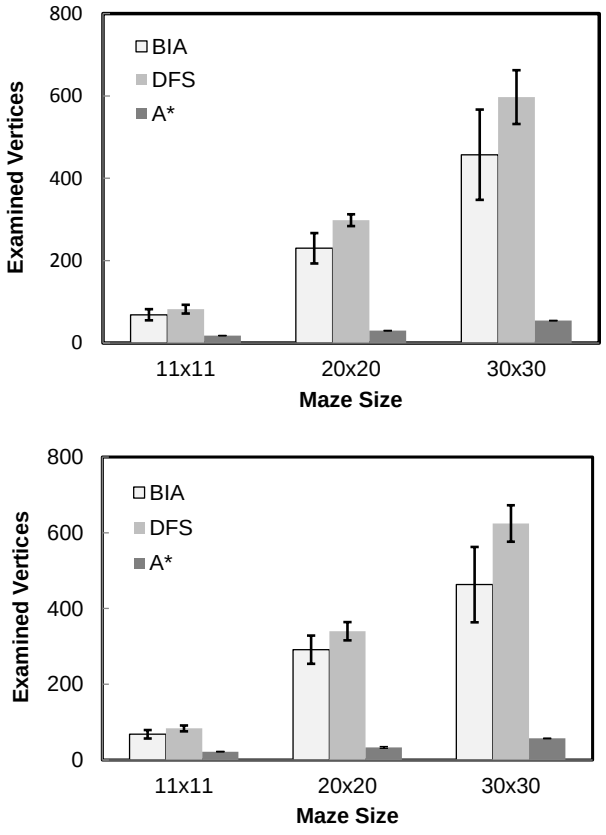

Fig 7. Reachable space, for non-randomized (top), and randomized (bottom) space, as a result of the exploration of mazes with various sizes by BIA, DFS and A* algorithms, respectively.

Running time. The comparison of the performance of the natural (BIA) algorithm with standard ones has been extended and deepened, by benchmarking the experimental running time

on a computer for uninformed algorithms, i.e., BIA, and DFS, and for informed maze search algorithms, i.e., Best First Search,[22] Jump Point Search,[23] and Dijkstra.[24] The tests have been run on a Dell Inspiron i3. The mazes used for tests were non-randomized.

Fig 8 (top) presents the computational performance of various maze-solving algorithms. For large mazes, the execution time increases significantly for uninformed algorithms, i.e., BIA and DFS. For uninformed algorithms, DFS performs slightly better in smaller maze size (up to 30x35). However, when the maze size keeps growing, BIA performs much better than DFS.

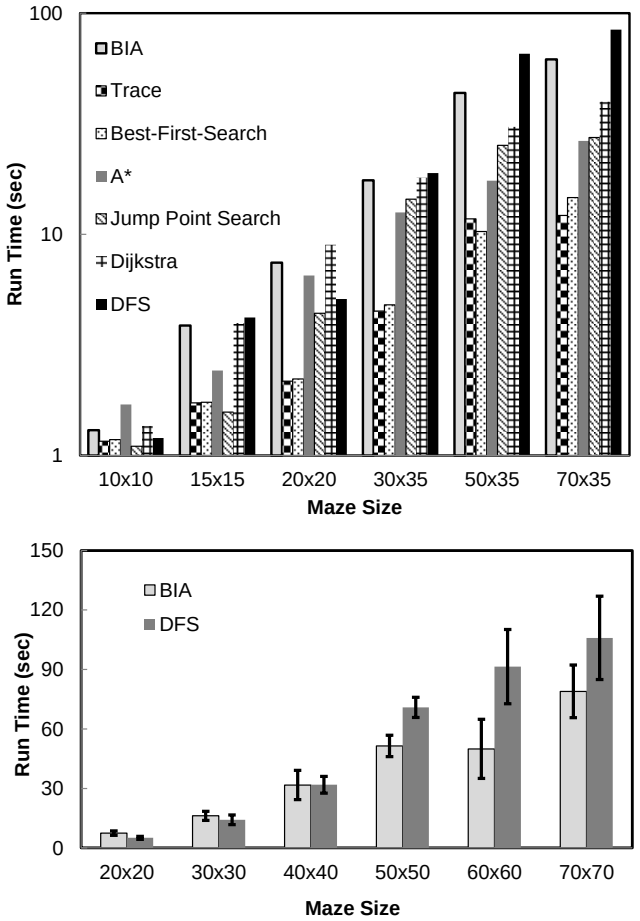

Fig 8. Evolution of the computing time vs. maze size for BIA and standard space search algorithms (top, logarithmic scale); and BIA and DFS (bottom)

A more 'correct' comparison, i.e., between the uninformed algorithms DFS and BIA, rather than comparing together the performance of informed and uninformed algorithms, demonstrates the efficiency of the natural algorithm (Fig 8, bottom). Interestingly, in smaller mazes (up to 30x30), DFS performs slightly better than BIA, but at mazes with sizes larger than 40x40, BIA performs 20-40% better (tested until 70x70).

Moreover, if we assume the dominant term of the running time is of the form run_time=an^b, where n is the maze size (as a product of the width and the height), we can derive the following asymptotic complexity of BIA and DFS (Table 1). Under these preliminary observations, the correlation between the computing time and the size of the maze predicts that the DFS algorithm will scale up with the size of the maze at a power ~2.4, whereas the Bio-Inspired Algorithm (BIA) will scale up at a power ~2.2. While the pre-exponential coefficient of the power law relationship between the computing time and the maze size is larger for BIA than that for DFS, the lower value of the exponent accounts for the better performance of BIA vs. DFS for larger mazes.

Table 1. Coefficients of run_time = a·(maze size)^b

Space searching algorithms	**a**	**b**	**R^2**
Trace	0.82	1.39	0.85
Best-First-Search	0.81	1.42	0.86
A*	1.25	1.61	0.94
Dijkstra	1.20	1.94	0.99
Jump Point Search	0.71	2.03	0.91
Bio-Inspired Algorithm	**1.00**	**2.19**	**0.96**
Depth First Search (DFS)	0.83	2.42	0.91

The above data indicate that, while BIA appears to scale up with the size of the maze worse than A* (a mid-performance algorithm within the class of informed ones), it is nevertheless not far away from some informed algorithms, such as Jump Point Search and Dijkstra. It needs to be noted that BIA, as implemented here, is a 'raw' algorithm, i.e., 'copied' from the natural one without any improvement. Consequently, future directions of research could comprise both exhaustive harvesting of natural algorithms, as well as their optimization, e.g., via an *in silico* evolutionary process.

Perspectives on natural space search algorithms. Many of the performance-related characteristics of the natural space searching algorithms, as well as those related to non-performance, can be understood when considering a biological perspective. Indeed, fungi have to survive with little or erratic distribution of nutrients, and a heterogeneous environment, without any means to 'estimate-ahead' these critical constraints. Furthermore, the variations in the space searching algorithms are also justified by species-specific environmental parameters. For instance, hard walls, e.g., rock or wood, require a collision-induce branching behind the leading tip, whereas soft walls, e.g., bread, require a split-at-the-top variant. Also, the overall structure of the confining environments, and their static or dynamic geometry, will modulate the 'length' of the directional memory. Finally, it appears that the fungal algorithms for space search bear some level of anthropomorphic characters, e.g., cooperation, competition, and the balance between them, even 'altruism' and 'sacrifice' for the benefit of the colony (and the species).

A brief analysis of the fungal algorithms also reveals interesting features, e.g., inherently stochastic behavior, caused by the variability of cellular responses to environmental constraints, decentralized 'computation' with local re-allocation of computational resources (i.e., biomass), the identical 'computational' nature of growing and moving, and an efficient balance between biological adaptability to immediate challenges versus 'grand plan' algorithmic design distilled during many evolutionary cycles.

These considerations suggest that an effort to harvest the biological space searching algorithms can be the basis for their reverse-engineering for graph and tree search strategies.

IV. Conclusions

We compared the performance of the natural algorithm for space search used by fungi, against several standard space searching algorithms, both uninformed of the maze structure, i.e., DFS algorithm, as well as informed ones, such as A*, Best First Search, Jump Point Search, and Dijkstra.

The fungal natural algorithm consistently outperforms DFS, and although it is inferior to other informed algorithms, e.g., A*, this under-performance does not increase with the increase of the size of the maze. This result suggests that further studies regarding the natural algorithms used by microorganisms with regard to their optimality vs. artificial counterparts are warranted, and perhaps overdue, and that they can lead to for their reverse-engineering for graph and tree search strategies.

Acknowledgements

Financially supported by the European Union the European Union Seventh Framework Programme (FP7/2007-2013, grant agreements 228971, MONAD; 613044, ABACUS) and Defense Advanced Research Projects Agency, grant agreement N66001-03-1-8913.

APPENDIX A:
Pseudo-code of the Bio-Inspired Algorithm (BIA)

1. Initialization:
   a. Create an array *UnvisitedCell* to store unvisited cell locations
   b. Start with an arbitrary direction to enter the maze
   c. Set *DirectionalMemory(1)* = the entering direction

Begin

   a. Move forward with the current direction until bump into the wall (go to Directional Memory Section) or corner (go to Corner-induced Collision Branching)
   b. Remove the visited cell locations away from *UnvisitedCell*

2. Directional Memory
   a. **If** cell collide on the wall
      Follow the wall direction
      End
   b. Once the cell leave the wall, follow the direction stored in *DirectionalMemory*
   c. **If** not applicable
      Select an arbitrary direction to restart
      Update *DirectionalMemory(n)* = the new direction
      End
3. Corner-induced Collision Branching
   a. **If** cell collide on the corner or the dead end
      Generate a new branch from UnvisitedCell
      Set *DirectionalMemory*(n) = the starting direction
      End

Repeat until the maze is solved

References

[1] H. Cho, H. Jönsson, K. Campbell, P. Melke, J. W. Williams, B. Jedynak, *et al.*, "Self-organization in high-density bacterial colonies: Efficient crowd control," *PLoS Biology*, vol. 5, pp. 2614-2623, 2007.

[2] D. Helbing, "Traffic and related self-driven many-particle systems," *Reviews of Modern Physics*, vol. 73, pp. 1067-1141, 2001.

[3] S. Binitha, S Siva, S, "A Survey of Bio inspired Optimization Algorithms," *International Journal of Soft Computing and Engineering*, vol. 2, pp. 137-151, 2012.

[4] K. L. Hanson, D. V. Nicolau Jr, L. Filipponi, L. Wang, A. P. Lee, and D. V. Nicolau, "Fungi use efficient algorithms for the exploration of microfluidic networks," *Small*, vol. 2, pp. 1212-1220, 2006.

[5] M. Held, C. Edwards, and D. V. Nicolau, "Probing the growth dynamics of Neurospora crassa with microfluidic structures," *Fungal Biology*, vol. 115, pp. 493-505, 2011.

[6] M. Held, A. P. Lee, C. Edwards, and D. V. Nicolau, "Microfluidics structures for probing the dynamic behaviour of filamentous fungi," *Microelectronic Engineering*, vol. 87, pp. 786-789, 2010.

[7] E. Korach and Z. Ostfeld, "Recognition of DFS trees: sequential and parallel algorithms with refined verifications," *Discrete Mathematics*, vol. 114, pp. 305-327, 1993.

[8] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, 1956.

[9] E. A. Wasserman and T. R. Zentall, *Comparative Cognition: Experimental Explorations of Animal Intelligence*, 2012.

[10] A. L. Nelson, E. Grant, J. M. Galeotti, and S. Rhody, "Maze exploration behaviors using an integrated evolutionary robotics environment," *Robotics and Autonomous Systems*, vol. 46, pp. 159-173, 2004.

[11] G. M. Viswanathan, S. V. Buldyrev, S. Havlin, M. G. E. Da Luz, E. P. Raposo, and H. E. Stanley, "Optimizing the success of random searches," *Nature*, vol. 401, pp. 911-914, 1999.

[12] P. Barthelemy, J. Bertolotti, and D. S. Wiersma, "A Lévy flight for light," *Nature*, vol. 453, pp. 495-498, 2008.

[13] A. M. Reynolds, "Deterministic walks with inverse-square power-law scaling are an emergent property of predators that use chemotaxis to locate randomly distributed prey," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 78, 2008.

[14] P. Erdos and A. Rényi, "On the strength of connectedness of a random graph," *Acta Mathematica Academiae Scientiarum Hungaricae*, vol. 12, pp. 261-267, 1961.

[15] S. S. Skiena, *The algorithm design manual: Second edition*, 2008.

[16] M. J. Fuerstman, P. Deschatelets, R. Kane, A. Schwartz, P. J. A. Kenis, J. M. Deutch, *et al.*, "Solving mazes using microfluidic networks," *Langmuir*, vol. 19, pp. 4714-4722, 2003.

[17] T. Nakagaki, H. Yamada, and Á. Tóth, "Maze-solving by an amoeboid organism," *Nature*, vol. 407, p. 470, 2000.

[18] S. Park, H. Hwang, S. W. Nam, F. Martinez, R. H. Austin, and W. S. Ryu, "Enhanced Caenorhabditis elegans locomotion in a structured microfluidic environment," *PLoS ONE*, vol. 3, 2008.

[19] S. Park, P. M. Wolanin, E. A. Yuzbashyan, H. Lin, N. C. Darnton, J. B. Stock, *et al.*, "Influence of topology on bacterial social interaction," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, pp. 13910-13915, 2003.

[20] S. Park, P. M. Wolanin, E. A. Yuzbashyan, P. Silberzan, J. B. Stock, and R. H. Austin, "Motion to form a quorum," *Science*, vol. 301, p. 188, 2003.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100-107, 1968.

[22] A. G. Pipe, T. C. Fogarty, and A. Winfield, "Balancing exploration with exploitation - Solving mazes with real numbered search spaces," in *IEEE Conference on Evolutionary Computation - Proceedings*, 1994, pp. 485-489.

[23] D. Harabor and A. Grastien, "Improving jump point search," in *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, 2014, pp. 128-135.

[24] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.