Human-Guided Enhancement of a Stochastic Local Search: Visualization and Adjustment of 3D Pheromone

Jaya Sreevalsan-Nair[1], Meike Verhoeven[1], David L. Woodruff[1], Ingrid Hotz[2], and Bernd Hamann[1]

[1] University of California, Davis; One Shields Avenue; Davis CA 95616; USA
`mverhoev, jnair, dlwoodruff, bhamann at ucdavis.edu`
[2] Konrad-Zuse-Zentrum für Informationstechnik; Takustrasse 7; D-14195 Berlin; Germany
`hotz at zib.de`

Abstract. In this paper we describe user interaction with an optimization algorithm via a sophisticated visualization interface that we developed for this purpose. The primary functions of our visualization tool are viewing the current data and interactively manipulating the available data to help in faster convergence of our iterative algorithm in a 3D environment with multiple obstacles and ant paths. We demonstrate that a user wielding this tool can improve the performance of an ant colony optimization algorithm as applied to a problem of finding 3D paths in the presence of impediments. Computational experiments demonstrate the value of this approach to engineering stochastic local search algorithms.

Keywords: ant colony optimization, shortest path, user in the loop, human guided search, visualization

1 Introduction

Interacting with an algorithm during execution can be a powerful way to facilitate engineering of local search algorithms and/or a way to improve their performance directly. In this paper we describe user interaction with an optimization algorithm via a sophisticated visualization interface that we created for this purpose. We demonstrate that a user wielding this tool can improve the performance of an ant colony optimization (ACO) algorithm as applied to a problem of finding 3D paths in the presence of impediments. There are a couple of reasons that we should investigate "user-in-the-loop" optimization. One is that a user might be able to interact with the algorithm find ways to improve the performance that can later be automated by finding generalizations of what the user does. In other situations, the nature of the problem instances may be changing so rapidly that a very general purpose algorithm (such as ACO) needs to be used and assisted by a user during each execution. In order to demonstrate how a user can improve an ACO algorithm, we develop a graphical tool and user interface enable the user to assist the algorithm.

Our problem is to find a shortest path between two points through a rectangular space that contains spherical impediments. Traveling through the spheres increases the length of the path. See [13] for a full description of the problem. In general, the problem cannot be solved exactly, so one solution method for this problem involves finding the shortest path through a rectangular grid. The more grid resolution, the better the solution. However, for three dimensional problems even the search for a shortest path through the grid requires computational effort that rises rapidly with the grid resolution. Hence, heuristic approaches are be needed.

Ant colony optimization (see, e.g., [4, 5]) is ideally suited to such a problem. The grid implies a network and every ant finds a path through it using a combination of greediness and pheromone left by previous ants. After an ant reaches the end it leaves some pheromone on every arc it used in proportion to the quality of the path: The shorter the path it has found, the more pheromone it leaves. Subsequent ants are more likely to use the arcs that have a higher amount of pheromone.

The improvement of the best path found by the ants is dependent on the luck and the experience of the other ants, but ants can not consider the structure of the problem. Since the structure of the problem is easy for a person to understand, humans have a good sense of how to improve ant-paths for example by deleting zigzagging that does not help avoiding any spheres. Therefore we need a good visualization tool that helps us see what the path created by the ants looks like and how we can give the ants hints to improve the result.

The visualization literature covers many aspects related to our problem. For example, analysis of search algorithm results can be effectively done using visualization [6]. Monitoring an algorithm using visualization gives the user all levels of information [3]. Visualization plays a major role in steering computations as can be seen in numerous cases. To list a few instances- visualization for interactive visual computing of programs [1], visualization in adaptive grid methods for debugging as well as analyzing computational algorithms [8]. Computational steering, which is the terminology for use of visualization to monitor simulations or iterations and using the feedback for improving computational results, has been a powerful research tool and a taxonomy of existing systems can be found at [10]. Mitsubishi research labs has conducted extensive research concerning what they call "human guided search" (see, e.g., [9, 7]). They have developed visualization tools for optimization problems such as scheduling, routing, and layout. One of their ideas is that computer algorithms can locate local minima given a starting point, but users can suggest new and better starting points. Our approach here continues this research line, but we provide tools for interacting directly with the operation of the ant system by allowing the user to lay down pheromone.

Visualization of, and interaction with, search paths in three dimension is not so easy, but we describe a tool in §3. Before we can do that we describe the problem and ACO algorithm in §2. In order to systematically demonstrate the potential value of having the user in the loop, we describe experiments in §4. The paper offers concluding remarks in §5

2 Problem formulation and Solution Method

A typical problem instance is shown in Figure 1. The goal is to get from the lower left corner to the upper right corner with the shortest possible path length, but the length is penalized when the path is inside one of the spheres.

2.1 Problem Formulation

A general formulation seeks the best path through a connected, closed set of points $\mathcal{A} \subset \Re^D$ from $a \in \mathcal{A}$ to $b \in \mathcal{A}$ where the presence of impediments affects the evaluation of the quality of the path. Since the original problem is, in general, not solvable, we create a finite set of points $\mathcal{G} \subset \mathcal{A}$, then for $i, j \in \mathcal{G}$ the path is given by the variable p_{ij} which is one if the arc from i to j is in the path and zero otherwise. The problem on a grid $\mathcal{G}$ is called (G) and is written as a shortest path problem:

$$\min_p \sum_{i \in \mathcal{G}} \sum_{j \in \mathcal{G}} \left[D_{ij} p_{ij} \left(1 + \sum_{s \in \mathcal{S}} c_s \beta(s, i, j) \right) \right] \qquad \text{(G)}$$

subject to:

$$\sum_{i \in \mathcal{G}} I_{ai} p_{ai} = 1$$

$$\sum_{i \in \mathcal{G}} I_{ib} p_{ib} = 1$$

$$\sum_{i \in \mathcal{G}} I_{ik} p_{ik} - \sum_{j \in \mathcal{G}} I_{kj} p_{kj} = 0, \quad k \in \mathcal{G} \backslash \{a, b\}$$

where the elements of incidence matrix, I_{ij}, have the value one if i and j are neighbors and zero otherwise; i.e., it indicates the presence of an arc between

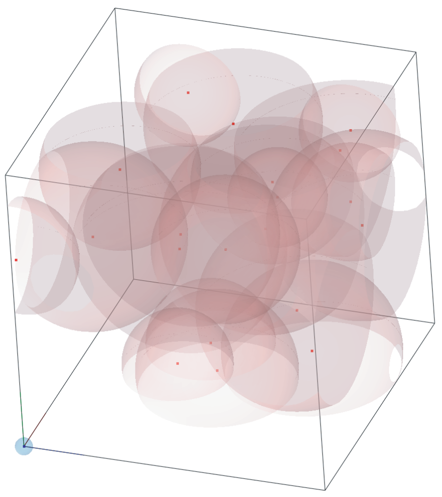


Fig. 1. Typical Problem Instance

grid points i and j. The elements of the distance matrix, D_{ij} give the distances between neighboring points i and j. This formulation assumes that a and b are both in the set $\mathcal{G}$. The data $\beta(s,i,j)$ gives the portion of the line segment between i and j that is in the range of impediment s. If the problem is to be solved using a general purpose solver, then these values must be pre-computed; however, if a shortest path algorithm is used, they can be computed as needed. The main point is that it is a shortest path problem so once the data is made available it can be computed rapidly for large instances. Unfortunately, for $D = 3$ the instances become *very* large even for modest resolution. We use a rectangular grid, where arcs are parallel to the coordinate axes meaning that the nodes only differ in one coordinate by $\frac{1}{k-1}$, diagonal in two dimensions meaning that they are connecting nodes that differ in two coordinates by $\frac{1}{k-1}$ or the three dimensional diagonal that means that nodes that differ in all three coordinates by $\frac{1}{k-1}$ are added.

2.2 Ant Colony

Each ant decides independently which node to go next to based on the pheromone trail τ_{ij} and a heuristic value η_{ij} for grid points i and j. We are searching for a path that consist of as few arcs as possible each of which is as short as possible, so we set our heuristic value so that it considers both aspects.

$$\eta_{ij} = \begin{cases} ((\frac{\sqrt{3}}{k-1} + |i-b| - |j-b|) * \gamma + \frac{1}{D_{ij}(1+\sum_{s\in\mathcal{S}} c_s \beta(s,i,j))} * (1-\gamma) & \text{, if } I_{ij} = 1 \\ 0 & \text{, otherwise} \end{cases}$$

where γ is a parameter on which aspect to emphasize more. The first term shows how much closer j is to the end point than i in Euclidean distance plus the maximal length of an arc, so that this value is positive. The second term gives the cost for this arc.

The ant located in the node i selects the arc to node j according to the probability

$$p_{ij}(t) = \frac{(\tau_{ij}(t))^\alpha (\eta_{ij})^\beta}{\sum_{n\in N} (\tau_{in}(t))^\alpha (\eta_{in})^\beta}, \forall j \in N$$

where the parameter α and β determine the relative influence of pheromone and heuristic value. After each ant reaches the endpoint, pheromone is deposited:

$$\Delta_{ij}(t) = \begin{cases} Q/L(t) & \text{if } (i,j) \in T(t) \\ 0 & \text{otherwise} \end{cases}$$

where Q is a constant and $L(t)$ is the length of the path $T(t)$ generated by ant t. So the shorter a path is the more pheromone is laid on its arcs. The amount of pheromone is updated according to the rule:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}(t)$$

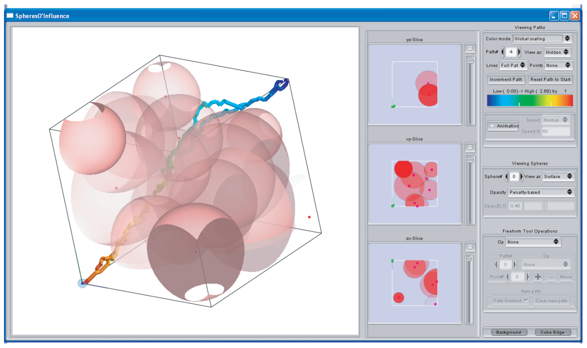
where ρ $(0 < \rho < 1)$ represents the persistence of the pheromone trail.

It is well known that some form of randomization is needed to improve the exploration done by the ants (see, e.g., [12, 2]). The scheme given by Nakamichi and Arita [11] has been shown to be affective and can be applied directly to our algorithm. We introduce a random selection rate r to improve the diversity of the paths. Since greedy strategies do in general not lead to the shortest path for our problem it is reasonable to increase the chances for arcs that are not that locally promising, but that may result in a better path. With a rate of r we replace our random distribution based on pheromone and heuristic by uniformly one over all arcs emanating from the node.

An important feature of our algorithm is the insertion of the user into the process. On arcs that are deemed to be useful by the user an extra pheromone amount of δ is added. This is usually done by "correcting" one of the paths found by an ant. In order to do that in three dimensions, a powerful visualization tool is needed. We confirm the value of this approach for our problem using experiments reported in §4.

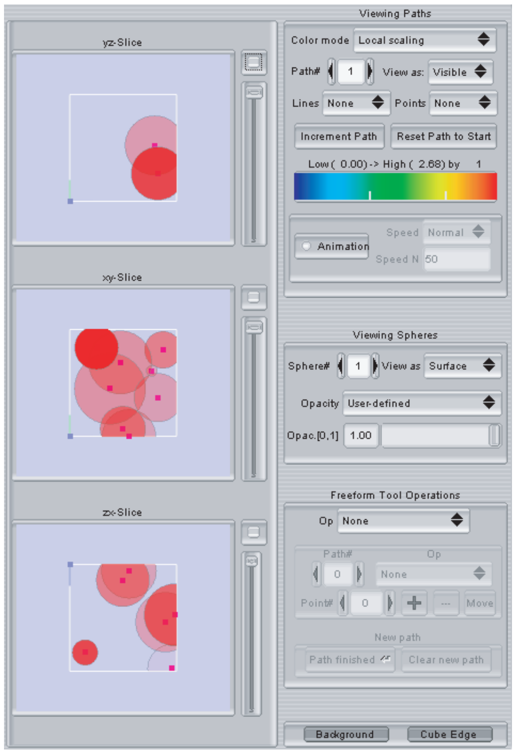
3 Visualization and Path Edit Tool

The primary functions of our visualization tool (shown in Figure 2) are viewing the current data and interactively manipulating the paths to help in faster convergence of our iterative algorithm in a 3D environment. The data to be visualized consists of paths, coordinates of the centers of spherical impediments, and their respective radii. These spheres are clipped by a bounding box, which is our volume of interest. Each sphere imposes a path-lenght penalty, which is indicated by the opacity of the sphere. As we generate paths using the ACO algorithm, they are plotted as thin solid tubes winding through the translucent spheres. Figure 3 zooms in on the controls that are shown in the right side of Figure 2.



Color mode: Global scaling
Path#: 4 View as: Hidden
Lines: Full Pat Points: None
Low (0.00) -> High (2.68) by 1
Speed: Normal
Speed N: 50
Sphere#: 0 View as: Surface
Opacity: Penalty-based
Opac.[0,1]: 0.40
Op: None
Path#: 0 Op: None
Point#: 0

Fig. 2. A snapshot of the complete visualization and interaction tool.

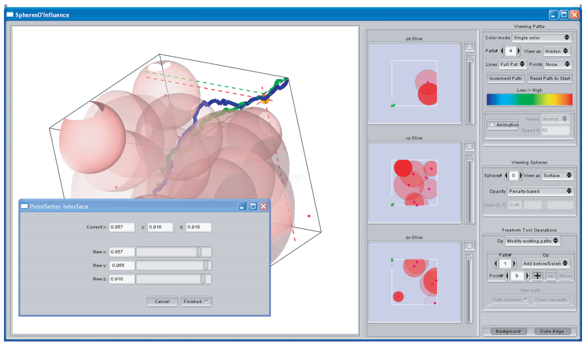


Color mode Local scaling
Path# 1 View as: Visible
Lines None Points None
Low (0.00) -> High (2.68) by 1
Speed Normal
Speed N 50
Sphere# 1 View as Surface
Opacity User-defined
Opac.[0,1] 1.00
Op None
Path# 0 Op None
Point# 0

Fig. 3. A snapshot of the controls.

Each path is represented as a vector of points. An exisiting path can be modified by inserting a point, deleting a point, or overwriting a point. The tool facilitates various features to interactively modify the paths, and save these changes to feed it back into the algorithm. Hence our tool includes an interface for modifying an existing point or adding a new point, as shown in Figure 4. This point modifying tool shows the current coordinates of the point and allows the user to move along the three orthogonal axes to place the point in a new position. In this snapshot, the modifications are taking place in the upper right corner.

The color of the path is determined by a by the cumulative (penalized) distance at each point on the path. The color is obtained by linearly interpolating a rainbow color scheme between zero and the maximum penalized distance possible. This allows us to have two different coloring possibilities- a global or a



Current x: 0.857 y: 0.918 z: 0.918
New x: 0.857
New y: 0.955
New z: 0.918

Color mode: Single color
Path# 4 View as: Hidden
Lines: Full Pat Points: None
Speed: Normal
Speed N: 50
Sphere# 0 View as: Surface
Opacity: Penalty-based
Opac.[0,1]: 0.40
Op: Modify existing paths
Path# 1 Op: Add before/Delete
Point# 9

Fig. 4. A snapshot of the visualization and interaction tool with the point modifying tool turned on and showing the two reference coordinates for the three axes and the point modifying interface overlaid on the main GUI.

local coloring scheme for each path. The global scheme makes use of extrema of penalties from all the paths, and the local scheme makes use of the penalties at the two ends of the chosen path. (The penalties at the ends of the path are the extrema, as we are considering cumulative, penalized distance here.)

For ease of visualization, the tool also gives different options for viewing the spheres: using the penalty-based opacity, using user-defined opacity or as wireframe. The tool gives navigational control to the user to rotate and translate the bounding box and its contents. Since this tool is used for research purpose, the interaction with the ACO algorithm is via a very simple, file based interface: The graphics tool can be viewed and tested by visiting `http://graphics.idav.ucdavis.edu/~jaya/InterdictionViz`.

To illustrate the improvement when interacting with the ants we stop the algorithm after the first 500 ants. We use our visualization tool to see which shortest path the ants have found so far and draw a path close to this which looks promising. We then add some pheromone on this path and continue the algorithm with another 500 ants. To compare this to the original algorithm we solve the same instance with 1000 ants. Since it takes some time and work for the user to create the hint we also let the ants solve the problem until they get the same shortest path as the ants with help. This gives us an idea of the advantage the hints give to the ants.

The user-in-the-loop process can be seen by looking at Figures 5 through 10, which traces the process for the network shown in Figure 1. The best path found by the first 500 ants is shown in Figure 6 and for the first 1000 without user intervention in Figure 7. However, if we add the hint shown in Figure 8 by modifying the best path and then laying down extra pheromone, we get a better path which is further improved by the next 500 ants as shown in Figure 9 and summarized in Figure 10. The same thing is summarized for another instance in Figure 11.

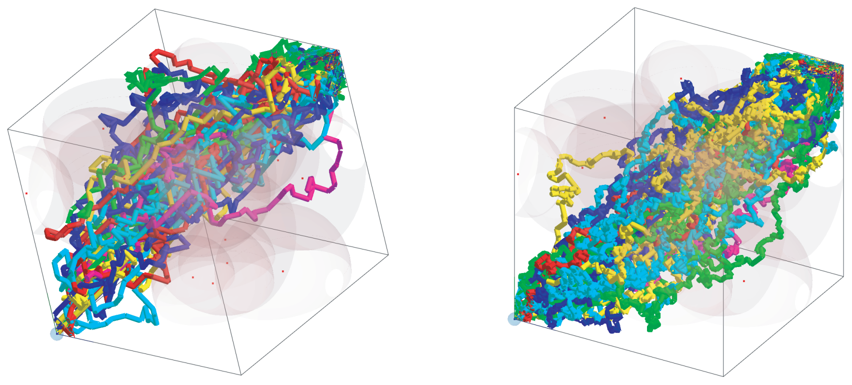


Fig. 5. Paths generated by the first 25 and 50 ants

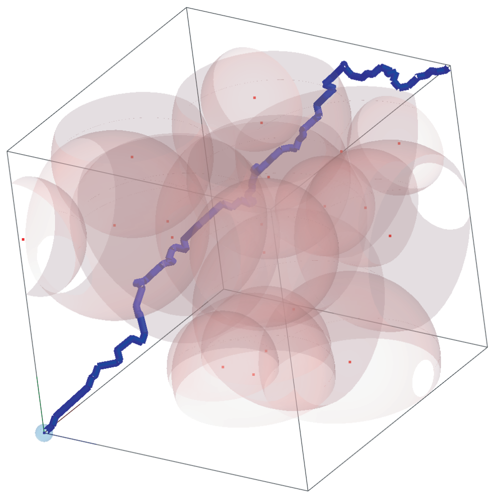


Fig. 6. Best path found by the first 500 ants

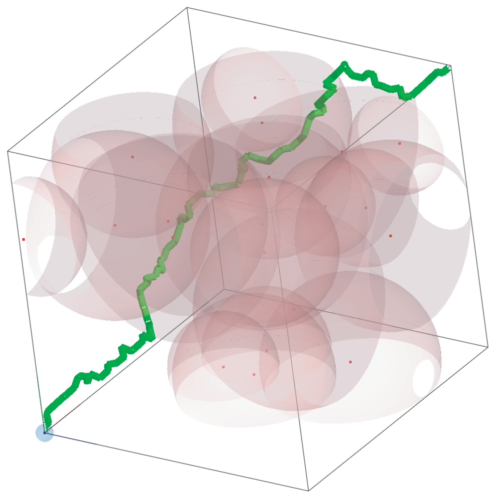


Fig. 7. Best path found by the first 1000 ants

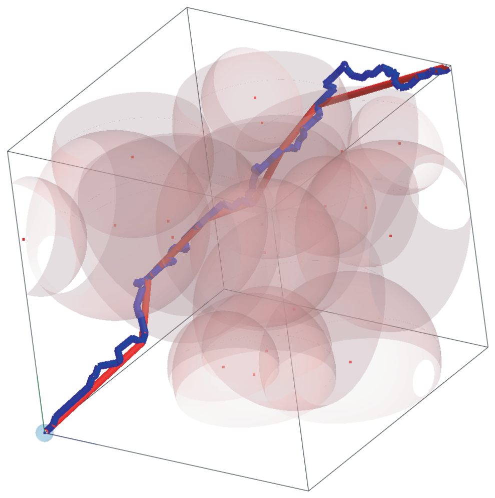


Fig. 8. Best path found by the first 500 ants and hint

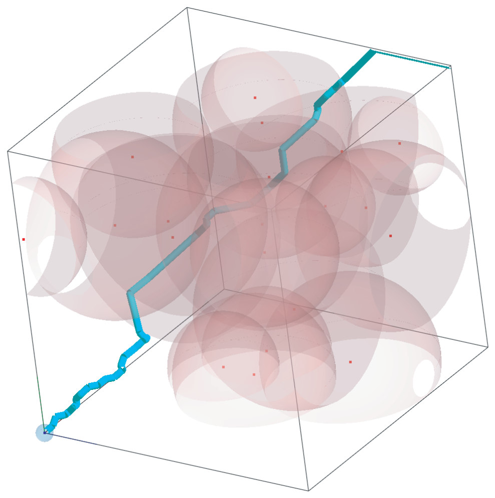


Fig. 9. Best path found by 1000 ants with hint

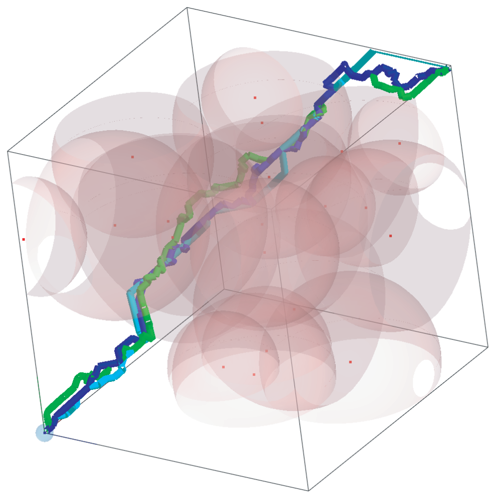


Fig. 10. Best path found by 500 ants and 1000 ants with and without hint

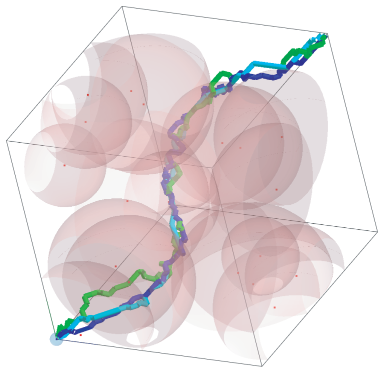
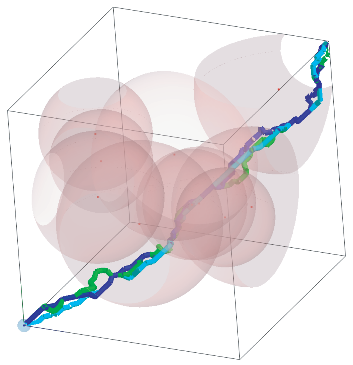



Fig. 11. Best path found by 500 ants and 1000 ants with and without hint

4 Experimental Results

To show the improvement of path developed by ants when the user is in the loop we generated six different problem instances. We found out that for these instances the ACO parameters $(\alpha, \beta, \gamma, \rho, Q, r, \delta) = (2, 3, 0.9, 0.99, 3, 0.03, 8)$ work well. Results are summarized in Table 1. All but the first and last columns show the best solution found after the given number ants. The column labeled "1000with" shows the results if a hint is given after 500 ants. The first column gives the instance number and the last column gives the number of ants needed to find a solution without a hint that is as good as the solution found with the hint. The hint is clearly quite valuable.

instance	500	1000with	1000	1500	2000	2500	ants until reached
1	2.880	2.180	2.546	2.283	2.224	2.170	2469
2	2.677	1.863	2.371	2.202	2.104	2.059	>500000
3	26.635	19.885	21.159	20.120	19.373	17.893	1643
4	8.544	5.935	7.514	7.113	6.908	6.854	11780
5	7.707	4.879	7.035	6.712	6.087	6.072	>500000
6	2.981	1.995	2.451	2.199	2.199	2.048	3545

Table 1. Results for the six instances

In order to demonstrate the need for the diversity parameter, r we conducted tests on the six instance with and without the parameter. Since ants with a higher chance of picking arcs randomly instead of taking arcs that are well known to give a short path are more likely to get paths that contain a high number of arcs, in average these ants take more CPU time to find the way through the network. Therefore in order to have a fair comparison while varying r, rather than comparing the length of the best path found by a certain number of ants we let the algorithm run for a fixed time. After 500 seconds we get the results shown in Table 2. Clearly, the randomization is needed. The effect is shown for one instance in Figure 12.

instance	r=0	r=0.03
1	2.395	2.751
2	3.094	2.298
3	16.996	21.004
4	9.622	7.426
5	6.072	5.681
6	4.413	1.941

Table 2. Results of a 500 second search conducted with, and without, randomization.

.

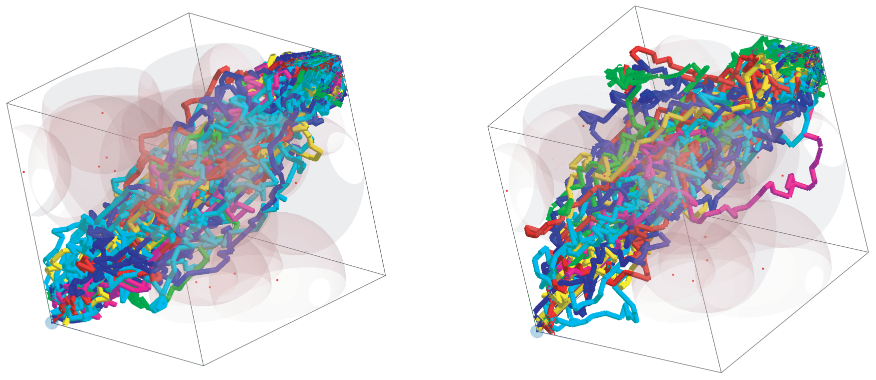


Fig. 12. First 25 ants for r=0 and r=0.03

5 Conclusions

We have described a problem of finding a path on a grid and an ACO algorithm to get good solutions to this problem. We also have described a powerful engineering tool for visualizing and interacting with the algorithm, which is challenging in a 3D environment. Ant systems are particularly well suited for visualization, but our main contribution has been to demonstrate the use of a tool for interacting with the algorithm by allowing the user to lay down pheromone and hence influence the future paths taken. Computational experiments demonstrate the value of this approach to engineering stochastic local search algorithms. The tool has been shown to result in improved performance of the algorithm by allowing the user to take advantage of their knowledge of the problem and convey that via the search control parameters. Human aided optimization is a fairly young topic and seems to offer substantial promise as part of efforts to engineer stochastic local search algorithms.

References

1. J. D. Brunner, D. J. Jablonowski, B. Bliss, and R. B. Haber. Vase: the visualization and application steering environment. In *Supercomputing '93: Proceedings of the*

1993 ACM/IEEE conference on Supercomputing, pages 560–569, New York, NY, USA, 1993. ACM Press.

2. B. Bullnheimer, R. Hartl, and C. Strauss. A new rank based version of the ant system. *Central European Journal for Operations Research and Economics*, 7:25–38, 1999.
3. Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
4. M. Dorigo and L.M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172, 1999.
5. M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26:1–13, 1997.
6. Simon P. Hammond. Putting the user in the loop: On-line user adaption of genetic algorithms. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 892–897, Canberra, 8-12 December 2003. IEEE Press.
7. G.W. Klau, N.B. Lesh, J.W. Marks, and M. Mitzenmacher. Human-guided tabu search. *National Conference on Artificial Intelligence (AAAI), ISBN: 0-262-51129-0*, pages 41–47.
8. O. Kreylos, A. M. Tesdall, B. Hamann, J. K. Hunter, and K. I. Joy. Interactive visualization and steering of cfd simulations. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, pages 25–34, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.
9. N. Lesh, L.B. Lopes, J. Marks, M. Mitzenmacher, and G.T. Schafer. Human-guided search for jobshop scheduling. *3rd International NASA Workshop on Planning and Scheduling for Space, October 2002*.
10. Jurriaan D. Mulder, Jarke J. van Wijk, and Robert van Liere. A survey of computational steering environments. *Future Gener. Comput. Syst.*, 15(1):119–129, 1999.
11. Y. Nakamichi and T. Arita. Diversity control in ant colony optimization. *Artificial Life and Robotics*, 7:1614–7456, 2004.
12. T. St utzle and H. Hoos. $\mathcal{MAX} - \mathcal{MIN}$ ant systems. *Future Generations Computer Systems Journal*, 16, 2000.
13. M. Verhoeven and D.L. Woodruff. Optimizing paths in the presence of spherical impediments, 2007.