Visualizing User Interactions with Simulation Tools

Nathan Denny
HUBzero
Purdue University
West Lafayette, IN, USA
ndenny@purdue.edu

Michael Zentner
HUBzero
Purdue University
West Lafayette, IN, USA
mzentner@purdue.edu

Gerhard Klimeck
Network for Computational Nanotechnology
Purdue University
West Lafayette, IN, USA
gekco@purdue.edu

***Abstract*—In order to improve user experiences with simulation tools hosted by cyberinfrastructure, we endeavor to gain a better understanding of how users interact with tools. The dimensionality of these tools is often too large to be intuitively understood. This paper presents two contributions to the study of user behavior: the MEANDER algorithm for visualizing sessions of user activity, and a scoring method ("searchiness") for characterizing a user's behavior along an axis of "wildcatting" vs. searching. The MEANDER algorithm uses graph heuristics to squash a high dimensional path of exploration into a (distorted) plane for rendering. The "searchiness" score is built upon the same graph techniques.**

***Index Terms*—cyberinfrastructure, user behavior, visualization**

I. Introduction

As part of our ongoing research and development of cyberinfrastructure, we are interested in investigating how our users interact with hosted simulation tools. By learning more about our users and how they use our hosted tools, we hope to enhance the users experience and ultimately improve upon how science is done with cyberinfrastructure.

Our present investigation began with the hypothesis that user interactions with tools could be broadly categorized along an axis of wildcatting to searching. This axis is analogous to the exploration vs. exploitation axis [1] [2] that is common in several fields of management and economics.

As the saying goes, "seeing is believing." To better understand user-tool interactions, we started by developing a visualization algorithm that could show how a user interacted with a particular tool. Our MEANDER algorithm uses graph theory and heuristics to model and render a users sequence of activity.

Using detailed activity logs from nanoHUB.org [3], we generated thousands of MEANDER plots. Once we could see user behavior and found several instances of the patterns we anticipated, we worked toward developing a method for automatically characterizing a users session behavior with respect to our wildcatting vs. searching axis.

This paper includes an overview of the MEANDER algorithm and our characterization method.

This material is based upon work supported by the Network for Computational Nanotechnology under US National Science Foundation grant nos. EEC-0228390, EEC-1227110, EEC-0228390, EEC-0634750, OCI-0438246, OCI-0832623 and OCI-0721680.

II. Nomenclature

A **tool** is an executable software object that embodies a simulation or model of some phenomenon of interest. In the case of nanoHUB, many tools will model nano-electronic systems or fine-grained material properties. A **tool user** is a registered nanoHUB user that interacts with a tool in the context of a tool session. A **tool session** is an instance of allocated compute resources, typically a lease on a virtualized machine that can last for a few minutes to a few days. While tool sessions are somewhat arbitrary, they often conceptually overlap with a particular purpose or question that motivates the tool user. Within the context of a tool session, a tool user can interact with the tool by issuing a query (or queries). A **query** is an abstraction of the inputs to the model executed by the tool, or the *domain* of the tools function. The query is answered by an invocation of the tool. An **invocation** of a tool is a single instantiated compute process that reads the query and produces a result. We assume tools are ideal lambda functions and thus have no state between invocations. The **result** of the invocation is an abstraction of the output of the tool, or the *range* of the tools function.

III. MEANDER

A MEANDER plot, such the one shown in Fig 1, is a visual representation of a tool users session. Each invocation is shown as a circle of various size and color. A line drawn between circles signals a sequential relationship, where the one invocation preceded another. The size of each circle reflects the degree of change in the inputs, relative to the preceding invocation. The color used to paint the circle is associated to a specific, distinct "edit set" of changed arguments.

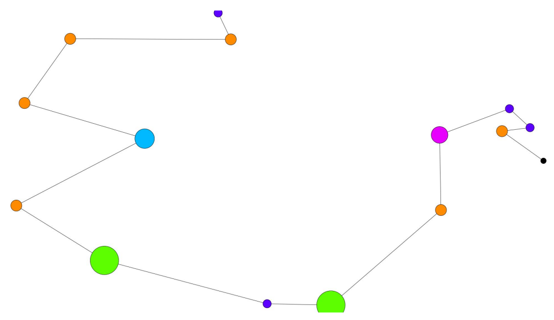
Fig. 1. MEANDER nanoHUB session 397899

A. Example

For example, assume we have a simple tool that calculates the volume of a rectangular solid. This example tool requires inputs dx, dy, and dz, all of which are lengths. Furthermore, assume a tool user that has, within a single tool session, invoked our example tool four distinct times, with the input arguments

query	dx	dy	dz
q_0	1	1	1
q_1	1	2	3
q_2	1	3	3
q_3	1	4	4
q_4	3	5	5

The vector q_0 is the vector of default values and is a special case that is the starting point in drawing the plot. If the tool user did not invoke q_0, this node is used in the plot algorithm, but will be otherwise invisible when drawn.

In executing query q_1, the user has (relative to the preceding argument vector, q_0) changed the arguments assigned to parameters dy and dz. The edit set is therefore $\{dy, dz\}$, and the size of the circle would be relative to the cardinality of the edit set (in this case, 2). The edit sets and edit distances for all queries are shown, below.

query	edit set	edit distance
q_1	$\{dy, dz\}$	2
q_2	$\{dy\}$	1
q_3	$\{dy, dz\}$	2
q_4	$\{dx, dy, dz\}$	3

The set of colors is the same size as the set of distinct edit sets, $\{\{dx, dy, dz\}, \{dy, dz\}, \{dy\}\}$. The specific mapping of color values to edit set is one-to-one, but otherwise arbitrary. In our example, we have assigned red to $\{dx, dy, dz\}$, blue to $\{dy, dz\}$, and gray to $\{dy\}$. We typically reserve black to color the initial default node, if it is visible. Figure 2 shows a possible MEANDER rendering of our example.

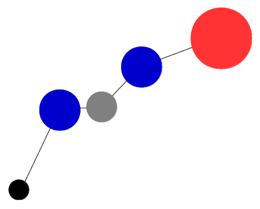

Fig. 2. MEANDER example plot

B. The Semantics of Distance

When comparing the arguments of two queries, we can construct a general distance function between the parameters that are scalar. So far, we have used a distance function that is the log-scaled difference between the two values. For our initial investigation, this method was sufficiently sensitive to both small and large changes in value.

Beyond scalar values, the semantics of distance are not clearly defined. What is the distance between True and False? What is the distance between the chemical elements gold (Au) and silver (Ag)?

If we know the context of the query, we might be able to construct a function that captures the specific semantics of distance. E.g. were we calculating momentum, the atomic mass might be a good proxy for the distance between gold and silver. However, in the general case, without any prior knowledge of the application, the concept of distance between discrete values is not obvious. For the results shown, we used the size of the option set as the distance between all discrete members in that set. This is somewhat arbitrary, but sufficient for our purposes, here.

In both scalar and discrete values, we have so far defined distances only for uni-dimensional comparisons. When the query is considered in total as a vector, we must have some reduction operation that produces a magnitude of distance. There are many common methods of computing topological distance. The "Manhattan" distance is the sum of the differences between the elements of the query vector. The "Chebyshev" distance is maximum distance between the arguments in the query vector. In our current work we use the generalized Euclidean distance …

$$\Delta d(p, q) = \sqrt{\sum_{i=1}^{m} (q_i - p_i)^2}$$

…where p and q are queries with m parameters.

C. Edit Sets and Edit Distance

In addition to the semantic distance between two queries, we are also interested in how many of the input arguments were changed from one invocation to the next. The edit set, (Δs) is the set of parameters that a tool user varied in query q_i, relative the preceding query, q_j.

$$\Delta e = |\Delta s(q_i, q_j)|$$

The "edit distance" (Δe) provides additional context in interpreting the semantic distance Δd between invocations. We have also contemplated edit distance as a coarse proxy for the cognitive effort invested by a tool user.

D. Algorithm to Construct MEANDER Plot

The algorithm for constructing a MEANDER plot starts with the set of input vectors extracted the user's session log. From these input vectors, we construct a complete graph (K_n) where each input vector is a vertex in the graph. Each edge in the graph is given an ideal length that is computed using a general distance function. The vertices and edges are decorated to visually represent changes made by the user in the sequence of invoking the tool.

Given a sequence of queries $Q = \{q_1, ..., q_n\}$

1) Construct a complete graph $K_n = \{V, E\}$ with each $q_i \in Q$ becoming vertex $v_i \in V$
2) Label each edge with an ideal length …

$$E(q_i, q_j) = \Delta d(q_i, q_j)$$

…for $q_i, q_j \in Q$, $i \neq j$

3) Decorate each vertex v_j by assigning the visual size of the vertex: $\Delta e(q_i, q_j)$
4) Decorate each vertex by asigning a color from the color map to each distinct change seen in the session.
5) Decorate edges, labeling edge $E(q_i, q_j)$ as visible if $\{q_i, q_j\}$ is a subsequence of Q. Label all remaining edges as invisible.
6) Serialize the graph structure into the "dot" language used by graphviz
7) Call graphviz' "neato" to perform the layout

E. Visualization and High Dimensionality

In our rectangular volume example, our queries had only three parameters. Visualizing the spatial distribution of the queries is straightforward for tools that have only 2 or 3 dimensions. Most of the tools available on nanoHUB.org are much more complex. Take for example, *nanofet*. This common tool on nanoHUB, has 53 numeric inputs and is not an unusual case. A sequence of queries to nanofet would be beyond our intuitive ability to visualize in 2D or 3D.

MEANDER renders a session with high dimesional queries using a heuristic that squashes the high dimensional structure into a plane. In general, it is not possible to compress high dimensional spaces into lower dimensional spaces without some distortion of distance. Therefore, the distances between nodes in MEANDER plots are representative of the semantic distance between the nodes, but should not be assumed to be of any particular consistent scale or transformation.

Following the construction algorithm, the MEANDER graph is fed into the graphviz [4] neato for layout. The neato layout uses a spring model where edges can be assigned an ideal length, but the layout can deform the length of an edge, if needed. The layout of a spring model graph is an optimization problem in minimizing the energy needed to deform the spring edges [5]. For our purposes, we treat the neato layout process as a black box.

IV. Characterizing Activity

MEANDER plots are dense in how much information is packed into the visualization. We needed a simpler metric that could be used to label the behavior seen in tool sessions and to index those sessions for storage and retrieval. A simpler metric could also be employed as input to reactive automation that would help shape the users experience in new, productive ways.

A. Searching vs. Wildcatting

From our nanoHUB MEANDER plots, we found many excellent examples of what we see as wildcatting behavior and searching behavior.

Wildcatting is a speculative survey of some space. Our intuition of wildcat behavior would be a sequence of relatively large moves over a large space. Wildcatting could be human behavior to get a feel for a space, or it could be the result of automation that is sampling a space to build a surrogate model. Wildcatting behavior can be seen in the MEANDER plot in Figure 3, where we see the user issuing queries that are relatively far apart while altering the same two or three arguments with each invocation.

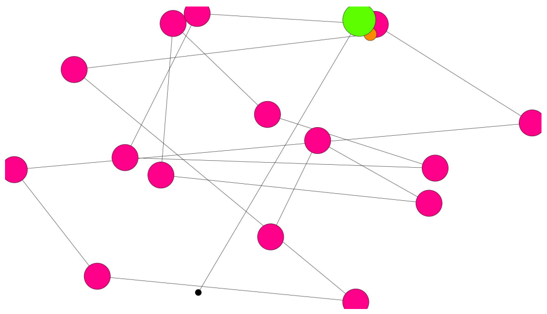

Fig. 3. MEANDER nanoHUB pntoy session 401554

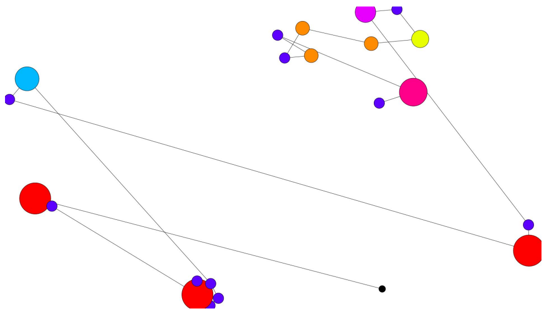

Fig. 4. MEANDER nanoHUB pntoy session 396749

In contrast to wildcatters, searchers are looking for local extrema. We would expect a searcher to move to some representative centroid point, then make relatively small adjustments in the topological neighborhood of the centroid. If those adjustments are not fruitful, we might see the user abandon the centroid, making a relatively large movement to a new point where the refinement process is repeated. We see searching behavior in Figure 4 where we find the user making several small movements in the neighborhood of relatively isolated centroid points.

B. "Searchiness" Metric

The MEANDER plot shows only the actual path of the tool user's activity. Beneath the MEANDER plot, the K_n graph contains potentially all possible activity paths. From our description of expected behavior of searchers and wildcatters, in general searchers will favor shorter movements in semantic space over longer movements. For the wildcatter we would expect almost the opposite behavior. The total distance of the shortest and longest (acyclic) path through the graph establishes an envelope of idealized, extreme behaviors. When we plot the actual tool user activity and the envelope boundaries, we can see in our examples, that the searcher stays close to the ideal path. The ratio of the area between the user path and the wildcat (upper bound) path to the area of the envelope gives us our "searchiness" metric. Note that the "searchiness" score will always be in the range $[0, 1]$; however we typically report these scores as a percentile.

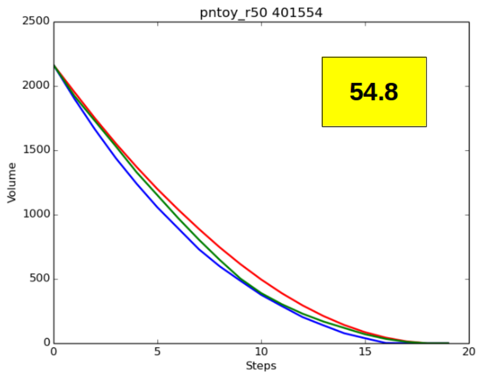

Fig. 5. MEANDER nanoHUB session 401554

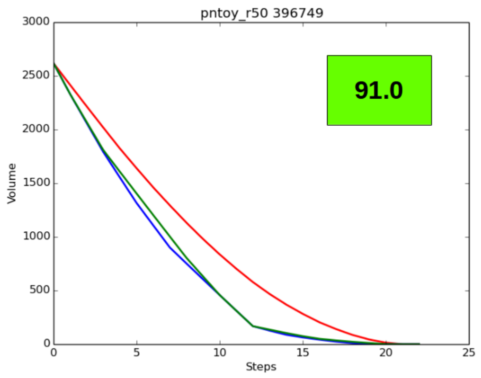

Fig. 6. MEANDER nanoHUB session 396749

Figure 5 shows the envelope and actual user activity path corresponding to the MEANDER plot in Figure 3. For this session, our "searchiness" computation yields a score of about 54%. An ideal wildcatter would have an activity path that tracked the upper bound of the envelope. In contrast, Figure 6, corresponding to the MEANDER plot in Figure 4, has a searchiness score of 91%. The searchiness scores in these two sessions are far enough aprt that we can confidently characterize them as exhibiting different behavior.

V. Applications and Future Directions

Our ultimate goal is to use our metrics to improve our users overall experience with nanoHUB. As of yet, we have no deployed applications that use MEANDER graphs or activity characterization. The following are two ideas that we have for the future application of MEANDER.

When a user invokes a tool, the corresponding query is first checked against the nanoHUB "Instant On" cache. Using a hash of the query's arguments, the cache can retrieve any matching, previoulsy computed results. Improvements to the Instant On service will use MEANDER methods to capture the gist of the user's session activity. The Instant On service would then answer a result that exactly matches the query as well as a set of suggested results that might be within the scope of the user's intentions.

A substantial fraction of our tool sessions are executed in the context of some academic course. If given a course roster that declares specific nanoHUB users as enrolled in an academic course, MEANDER visualization and characterization could help an instructor see how the students are approaching solutions to assigned problems. By seeing the search paths or wildcat fields of each student, the instructor could make ongoing fine-grained adjustments to the content of the course. MEANDER would then be part of a larger system that provides highly targeted and individualized learning.

VI. Conclusion

Our MEANDER methods visualize and characterize user interactions with simulation tools. MEANDER squashes a path of activity through high dimensional space into a 2D plot that can show how a user explores or surveys the domain of the tool. Our "searchiness" metric characterizes user activity onto an axis with wildcatting and searching at opposite extremes. The two full examples given in this paper show the visualization and characterization applied to actual logs of user activity. These examples demonstrate the "searchiness" characteristic differentiates our example wildcatter and searcher sessions. We intend to publish additional details and findings in an expanded version of this paper.

Acknowledgment

The authors would like to thank the Network for Computational Nanotechnology (NCN) for contributing data from nanoHUB.org The authors would also like to specifically thank Tanya Faltens and Lynn Zentner of the NCN for their thoughtful comments and suggestions while developing the MEANDER methods.

References

[1] J. G. March, "Exploration and Exploitation in Organizational Learning," *Organization Science*, vol. 2, no. 1,, pp. 71–87, 1991. [Online]. Available: http://www.jstor.org/stable/2634940
[2] A. K. Gupta, K. G. Smith, and C. E. Shalley, "The Interplay between Exploration and Exploitation," *The Academy of Management Journal*, vol. 49, no. 4, pp. 693–706, 2006. [Online]. Available: http://www.jstor.org/stable/20159793
[3] Madhavan Krishna, Zentner Lynn, Farnsworth Victoria, Shivarajapura Swaroop, Zentner Michael, Denny Nathan, and Klimeck Gerhard, "nanoHUB.org: cloud-based services for nanoscale modeling, simulation, and education," *Nanotechnology Reviews*, vol. 2, no. 1, p. 107, 2013. [Online]. Available: https://www.degruyter.com/view/j/ntrev.2013.2.issue-1/ntrev-2012-0043/ntrev-2012-0043.xml
[4] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *Software - Practice and Experience*, vol. 30, no. 11, pp. 1203–1233, 2000.
[5] E. R. Gansner, Y. Koren, and S. North, "Graph Drawing by Stress Majorization," in *Graph Drawing*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sep. 2004, pp. 239–250. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-31843-9_25