SAND2015-2324R

Visualization for Hyper-Heuristics: Front-End Graphical User Interface

Lauren Kroenung

Faculty Advisor:
Dr. Daniel Tauritz
Department of Computer Science

March 28, 2015

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Abstract

Modern society is faced with ever more complex problems, many of which can be formulated as generate-and-test optimization problems. General-purpose optimization algorithms are not well suited for real-world scenarios where many instances of the same problem class need to be repeatedly and efficiently solved because they are not targeted to a particular scenario. Hyper-heuristics automate the design of algorithms to create a custom algorithm for a particular scenario. While such automated design has great advantages, it can often be difficult to understand exactly how a design was derived and why it should be trusted. This project aims to address these issues of usability by creating an easy-to-use graphical user interface (GUI) for hyper-heuristics to support practitioners, as well as scientific visualization of the produced automated designs. My contributions to this project are exhibited in the user-facing portion of the developed system and the detailed scientific visualizations created from back-end data.

1 Introduction

Many practical problems cannot be solved using exhaustive search methods. With exhaustive methods, it would simply cost too much time to find optimal solutions. For such problems heuristics can be beneficial. Heuristics are incomplete search methods that do not ensure finding the most optimal solution [1]. However, they can get close to optimal in much shorter times. Hyper-heuristics automate the design of algorithms by combining multiple heuristics in order to create a custom algorithm or solution for a particular scenario.

Though hyper-heuristics can be efficient, it is hard to figure out why a resulting solution is more favorable over others. Since it is also difficult to discern how the system arrived at a specific solution, practitioners can have a hard time understanding why they should use that solution. Without any tools to analyze how the hyper-heuristics came to a particular solution or what makes that solution better than others, a practitioner could potentially abandon the results altogether in favor of a lesser solution of known origin.

The goal of this research is to provide those missing tools and allow users an interactive, in-depth analysis of the evolved algorithms and their experimental results. This project involved the creation of a user-friendly GUI that has the ability to integrate with an arbitrary hyper-heuristic driven framework and then visualize the data produced. The developed user interface can provide interactive visualizers with which to analyze post-run results and can also be used to modify parameters of the integrated framework.

Hyper-heuristics can be applied to a number of different search methods, but for this research we were focused on hyper-heuristics employed with genetic programming (GP), which is a type of evolutionary algorithm (EA). In the field of EA visualization, there has been research in both visualizing the genetic history of individual solutions in order to determine how a solution came to be [2] as well as visualizing population information and genetic lineage from GP runs [3]. However, none so far has dealt with both hyper-heuristics and EAs in the same interface. While some interactivity exists in prior experimental applications, this research project aims to have more.

In this project, two important aspects of GP with hyper-heuristics were focused on for creating visualizations: the individuals being evolved and phylogenetic trees that are generated post-run from the framework. The hyper-heuristic framework that was worked with during the semester, HYDRA, specifically dealt with evolving Black Box Search Algorithms (BBSAs) [4] [5] [6].

2 Design Decisions

2.1 Deciding on a Language

It was decided early on that the application should work on a user's machine regardless of their operating system; therefore it needed to be cross-platform. The language used to write the user interface needed to be chosen before any development could begin. Three solid options with their corresponding advantages and disadvantages were discussed. Figure 1 shows an overview of each language that was considered.

If implemented as a web application, the only prerequisite for running the interface system was having an Internet browser installed. This would make the application easily independent of a user's operating system and it could be designed responsively to work on any screen size, allowing for mobile support in the future. While using Python and Tkinter would allow for cross-platform compatibility, the user interface would not have been able to run on mobile devices such as phones or tablets. Once this was realized, the Python and Tkinter option was discarded.

Both HTML5 and Java could be employed within a web browser, however, there were more disadvantages associated with using a Java applet. Most computers do not come preinstalled with Java, which is what a Java applet needs to run. This means a user would have to download Java before being able to use the interface, if they did not have it installed already. Even then, all browsers do not support Java applets. With these limitations, it was clear that if browser compatibility was the goal then HTML5 was the final answer. The only downside to this option is that older browsers may not support the HTML5 canvas element and there are a small percentage of users that do

Figure 1: Language Options		
Language	**Pros**	**Cons**
HTML5 (Canvas + JavaScript)	• Not dependent on user's operating system • A very small percentage of internet users have JavaScript disabled • Variety of interactive visualization libraries available • Could use Python as app's backend with Django	• Older browsers may not support Canvas • HTML5 Canvas is stateless, must track everything manually • Canvas is redrawn every time an element moves, which can slow the browser
Python (Tkinter)	• Will work natively on Windows, Mac, and Linux • Able to quickly build full application • The hyper-heuristic framework being integrated with was already written in Python	• Very generic look • Would take effort to make anything user-friendly • Harder to involve interactive visualizations • Not able to do web application
Java (Applet)	• Could still implement as a web application • Therefore not dependent on user's operating system • Java is useful when it comes to drawing graphics	• Not supported by all browsers • Java does not come installed by default on many machines • Can be slow to initialize

Figure 1: Table of the languages considered to build the interface in.

not have JavaScript enabled on their browsers. Overall however, HTML5 would work for a larger proportion of potential users than a Java applet would.

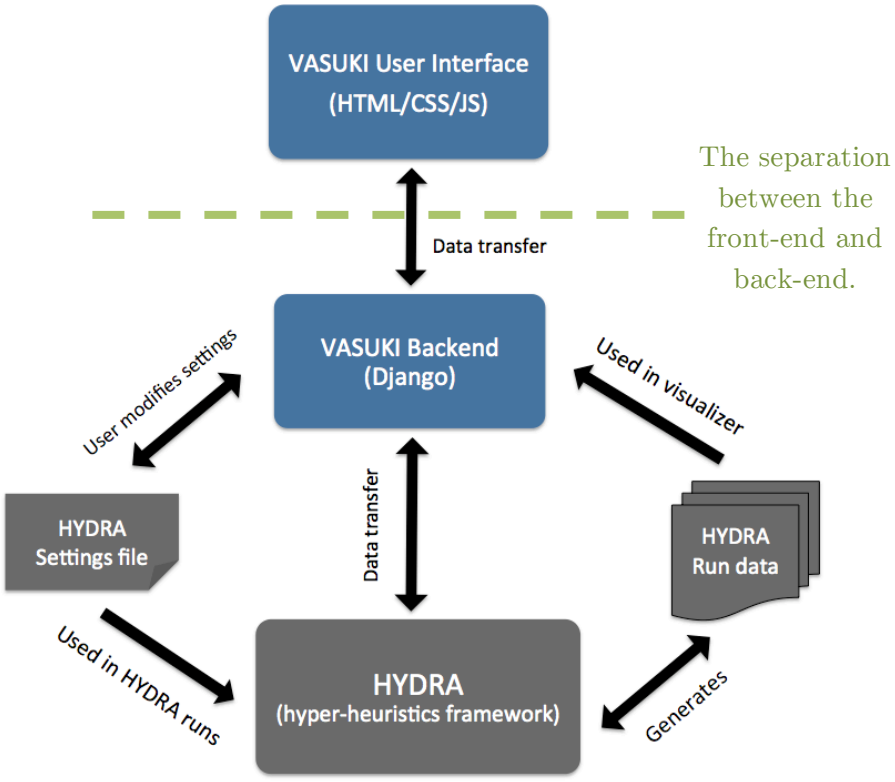
2.2 Hyper-heuristic Framework Flexibility

It was known from the beginning that the resulting user interface should work with an arbitrary hyper-heuristic framework running beside it. The interface application needed to be designed with that aspect in mind so it could be compatible with a variety of different hyper-heuristic driven frameworks, allowing for future growth. In order to accommodate this characteristic, the various pages in the GUI are dynamically generated from the settings file provided by the current framework. Choosing to build pages dynamically means that when the interface needs to integrate with a different hyper-heuristic framework, the only action necessary is to feed it a different settings file from that framework.

3 Implementation

There were two distinct parts in implementing this user interface, which was given the name VASUKI (Visualization and Scientific User Kontrol Interface). It was necessary to build both a front-end as well as a back-end for the web application to function. The back-end was constructed using Django (a Python web framework) and it was the part of the application that integrated with the hyper-heuristic framework. All the functionality employed within Django involves parsing and manipulating data retrieved from the framework, which is then sent to the front-end. The front-end was written in HTML5, CSS, and JavaScript along with the ArborJS library. This portion of the application involved the look and feel of the user interface, the format of the dynamically generated pages, and the interactive visualizer that displayed post-run data from the framework. As seen in Figure 2, there was a clear split in the work that had to be done for this project, so I contributed to the front-end coding, while another undergraduate researcher focused on the back-end.

Figure 2: Flow of the web interface



3.1 User Interface Design

The final layout design of the interface was structured responsively to be able to fit within the browser's viewport without any unseemly breaking of page elements. The

interface can be viewed from a range of devices, including phones, tablets, laptops, and desktop computers. Full system functionality while on mobile devices has not been implemented from the back-end; however, the front-end is coded to handle browser window resizing as elegantly as possible.

3.2 Dynamically Generated Pages from Settings File

The settings file provided by the hyper-heuristic framework is parsed in a way where pages that allow users to edit run settings can be dynamically generated. When the user modifies the settings values on the interface, they are saved into the actual settings file used by the framework while it runs. By having this aspect built into the GUI, users can edit the framework's settings, tell the framework to start a run, and then analyze that run's results all in the same interface.

As it can be difficult for practitioners to understand variable names that stem from the framework's given settings file, tooltips have been incorporated into these pages. These tooltips appear whenever the user's mouse hovers on top of an input field's label to further explain what this particular settings field modifies within the framework.

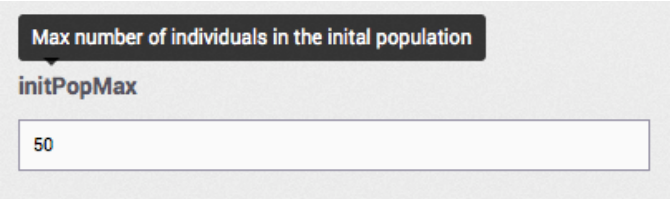


Figure 3: a tooltip on a settings page

3.3 Phylogenetic Tree Visualizer

When looking at the phylogenetic visualizer, each node on the tree represents one individual in the GP process. The directed edges represent gene inheritance from one individual to another. An example of this can be seen in Figure 4. For the HDYRA framework, if a user clicks on one of the nodes in the phylogenetic tree visualizer, they can view the corresponding individual BBSA tree for further analysis of that solution's genes [4]. Viewing the individuals like this can assist practitioners in understanding what parts of a solution make it better than others.

Users can comprehend an individual's genealogy by viewing the phylogenetic tree as a whole. Within a BBSA tree, the ability to view the originating individual of any given gene found is also provided. Being able to see an individual gene's ancestry can help users understand how the framework arrived at a particular solution.

One method available in determining the validity of a solution produced by hyper-heuristics is to trace its origin. By investigating the origin of certain genes in an individual, a practitioner is able to determine why a particular set of genes was passed on. If a set of genes survives several generations, this implies that this particular set is likely to be advantageous to the solution.

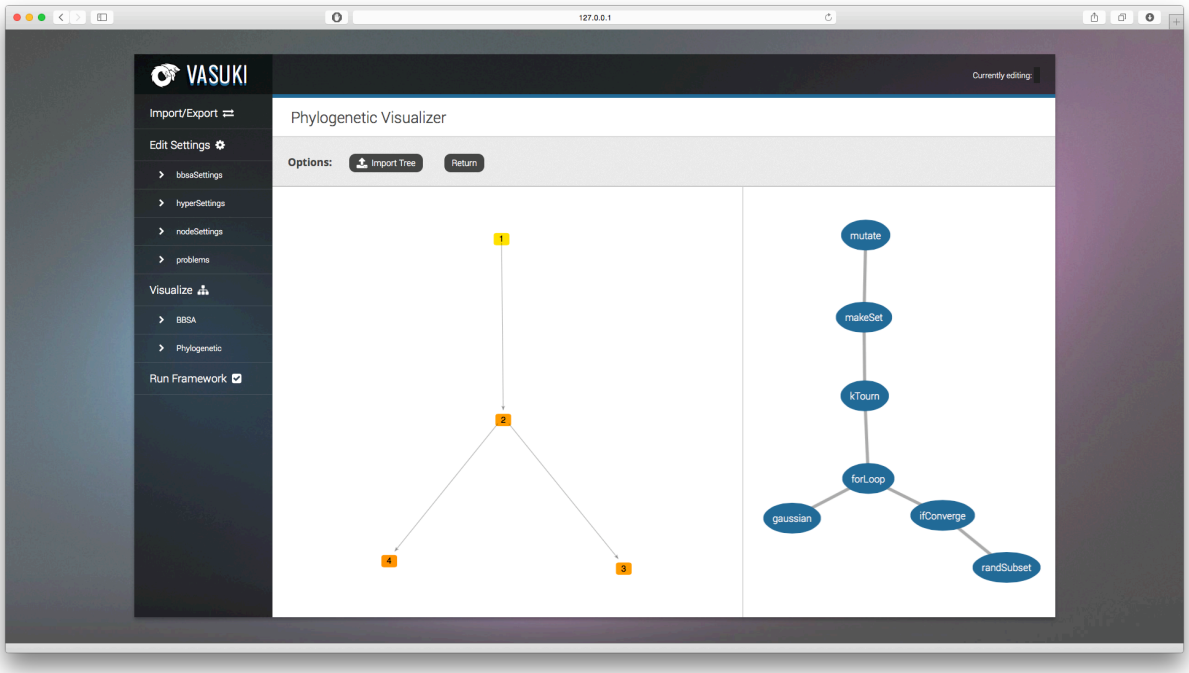


Figure 4: Phylogenetic tree on the left, one of the individuals on the right

3.4 Individual Visualizer

This visualizer displays individuals respective to the integrated hyper-heuristic framework. With the HYDRA framework, the individuals within its GP are tree-represented BBSAs [4]. An example of this can be seen in Figure 5. The individual visualizer has two functionalities within the web interface: (1) it can be used to create de novo individuals for use in the initial population of the framework's GP and (2) the same visualization renderer is also utilized when analyzing specific individuals from the resulting hyper-heuristic framework run data, where the run data is displayed as a phylogenetic tree.

4 Conclusion

Runs performed in hyper-heuristic frameworks produce complex data that can be difficult to interpret by practitioners. The visualizations produced by this interface give users the ability to perform interactive, in-depth analyses of the evolved algorithms and their experimental results. Such analyses can assist practitioners and researchers in uncovering the origin of particular solutions and what makes those solutions better than others. The strengths of VASUKI lie in the interactive characteristics of its visualizers. The VASUKI interface will be beneficial in aiding the understanding of a variety of hyper-heuristic frameworks, as this experimental system is able to function independent of where the data originates.

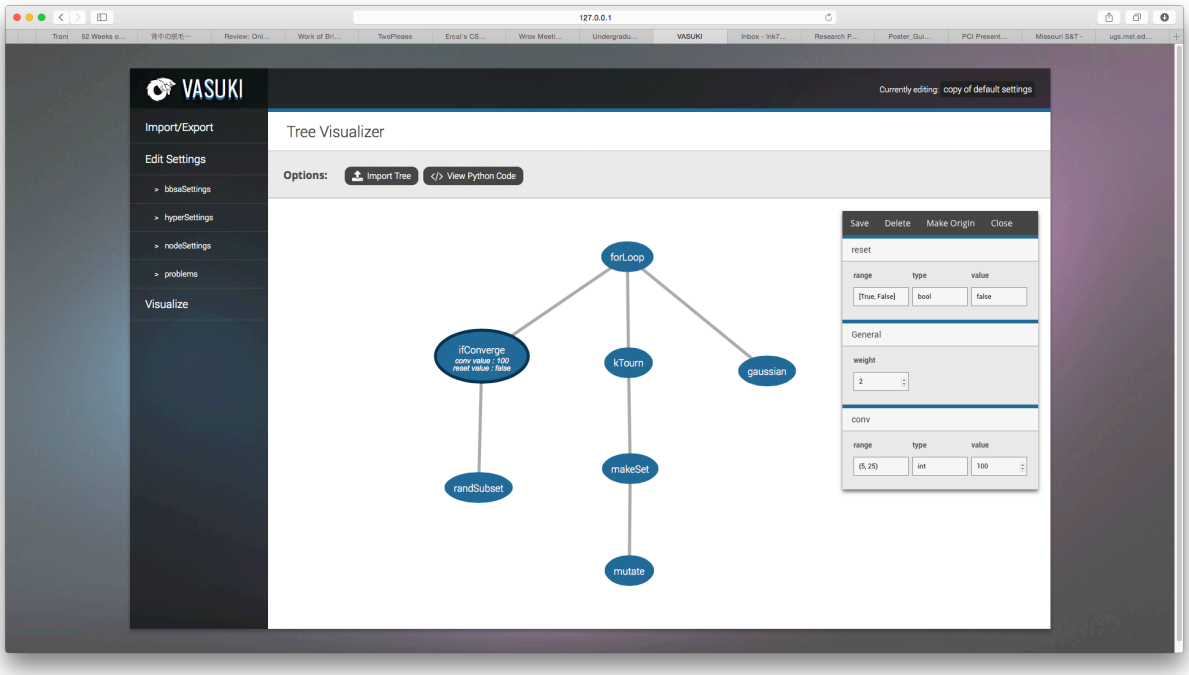


Figure 5: a BBSA tree in the individual visualizer

5 Acknowledgments

I would like to acknowledge and thank Dr. Daniel Tauritz, Matt Martin, and Luke Simon for all their guidance and support during this project.

References

[1] Ross, P. (2005) 'Hyper-Heuristics', in Ross, P., Burke, E.K. and Kendall, G. (ed.) Search Methodologies. US: Springer, pp. 529-556.

[2] E. Hart and P. Ross, "GAVEL - A new tool for genetic algorithm visualization", IEEE Transactions on Evolutionary Computation, vol.15, no.4, pp. 335-348, Aug 2001.

[3] Bogdan Burlacu, Michael Affenzeller, Michael Kommenda, Stephan Winkler, and Gabriel Kronberger. 2013. Visualization of genetic lineages and inheritance information in genetic programming. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation* (GECCO '13 Companion), Christian Blum (Ed.). ACM, New York, NY, USA, 1351-1358.

[4] Matthew A. Martin and Daniel R. Tauritz. 2013. Evolving black-box search algorithms employing genetic programming. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation* (GECCO '13 Companion), Christian Blum (Ed.). ACM, New York, NY, USA, 1497-1504.

[5] Matthew A. Martin and Daniel R. Tauritz. 2014. A problem configuration study of the robustness of a black-box search algorithm hyper-heuristic. In *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion* (GECCO Comp '14). ACM, New York, NY, USA, 1389-1396.

[6] Matthew A. Martin and Daniel R. Tauritz. 2014. Multi-sample evolution of robust black-box search algorithms. In *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion* (GECCO Comp '14). ACM, New York, NY, USA, 195-196.