# Parameter Sweeps for Exploring GP Parameters

Michael E. Samples, Jason M. Daida, Matthew Byom, Matt Pizzimenti
Center for the Study of Complex Systems and the Department of Atmospheric, Oceanic, and Space Sciences
University of Michigan
Ann Arbor, MI 48109
{msamples, daida}@umich.edu

## Categories and Subject Descriptors
I.6.7—Simulation Support Systems Environments; I.2.2—Automatic Programming Program Synthesis

## General Terms
Experimentation, Algorithms, Performance

## Keywords
Parameter Sweep, Experiment Management, Evolutionary Computation, Distributed Computation, Data Reduction.

Invoking a genetic programming (GP) engine to solve a particular problem can be thought of as performing a search on the sample space of all programs expressible in a given language of functions and terminals. Results obtained from such a search are the product of many low-level nonlinear interactions, the rules of which are highly dependent on engine configurations. With only slight changes, different configurations can produce dramatically different results. It is the experimentalist's goal to understand how specific configurations influence results, but it is impossible to exhaustively test all possible configurations. While small experiments can lead to knowledge about a given configuration's behavior, the results are difficult to other configurations due to GP's nonlinear interactions. As a result, empirical knowledge of GP's behavior can only be constructed by testing on multiple classes of configurations. Theories capable of encapsulating the varied behaviors can then be constructed and used to predict behavior with untested configurations. *Succinctly, this means that it is critical to test for an observed phenomenon by conducting multiple trials and changing input parameters. Experimentalists should be prepared to do this*. We call an experiment that involves the sampling of multiple parameter configurations a *parameter-sweep experiment* (PSE). This methodology has often been practiced in other fields (e.g., agent-based modeling) but is rarely practiced in our field. We suggest that the number of large multi-configuration studies is relatively small because PSEs are often prohibitively difficult for researchers to conduct. We present **Commander** – a tool in performing large *generic parameter-sweep experiments* in grid and cluster computing environments. ***Commander is not a GP system, but rather a software utility that performs automated experiment management and data reduction for PSEs.***

We identify three tasks that make PSEs difficult to conduct and explain how Commander approaches their solutions. In presenting these solutions, emphasis is placed on giving the user necessary control but hiding unnecessarily tedious tasks.

- *Experimental Setup* – Small experiments involving a single configuration class of parameters (i.e. one datapoint) can easily be conducted by writing a shell script capable of executing a GP program a number of times with the same inputs. Increasing the number of datapoints increases the complexity of the scripts because varied inputs must be sent to the GP engine. Satisfactory scripts are generally lengthy and non-portable—meaning that the construction of a new experiment involves large time-consuming changes to the script. Competent GP researchers should not need to have expert skills in writing these scripts. Commander solves this problem by providing a language for users to describe the parameters over which to sweep (e.g., popsize = {100,200,300}; generations = {50,100} ). Commander interprets a user's *experiment description file* and automatically conducts all associated experiments. The experiment description file is easy to modify, allowing experimentalists to rapidly define and run complex PSEs.
- *Required CPU hours* – Evaluating a single combination of parameters can be quite CPU-intensive, as GP is inherently nondeterministic and can require the reevaluation of the same configuration many times for statistically significant results. PSEs amplify this tendency since they evaluate a larger number of configurations. Parallel processing can ameliorate this effect, but increases the complexity of scripting. These scripts must focus on distribution algorithms (pairing jobs with available machines), job robustness, and data collection—requiring skills that competent GP researchers may not possess. Our Commander provides transparent support for parallel processing on grid and cluster networks. This process is described in more detail below.
- *Data Reduction* – Single datapoint experiments all produce the same type of data inasmuch as all data is generated by the same configuration. As a result, the extraction of relevant information is a straightforward process. In PSEs, the extraction of relevant information necessitates the use of scripts to arrange the data in meaningful ways. Also, when sweeping across multiple parameters there are complications in tagging data sets with appropriate configuration labels. These analysis scripts are difficult to maintain between large experiments as the configuration spaces change. Commander automatically applies predefined engine-specific *analysis functions* for data reduction and information interpretation. For example, one analysis function that comes with MGP (our GP engine), calculates average fitness over a domain of an arbitrary set of trials. When performing an experiment, users can select the average fitness analysis function to be paired with independent variables from the experiment description file's parameters. Commander transparently partitions the results by those variables and constructs graphs of average fitness as a function of the given independent variables. This leads to low turn-around time between experiments and thus is particularly valuable for GP experimentalists as it helps encapsulate the overwhelming


*GECCO'05,* June 25-29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

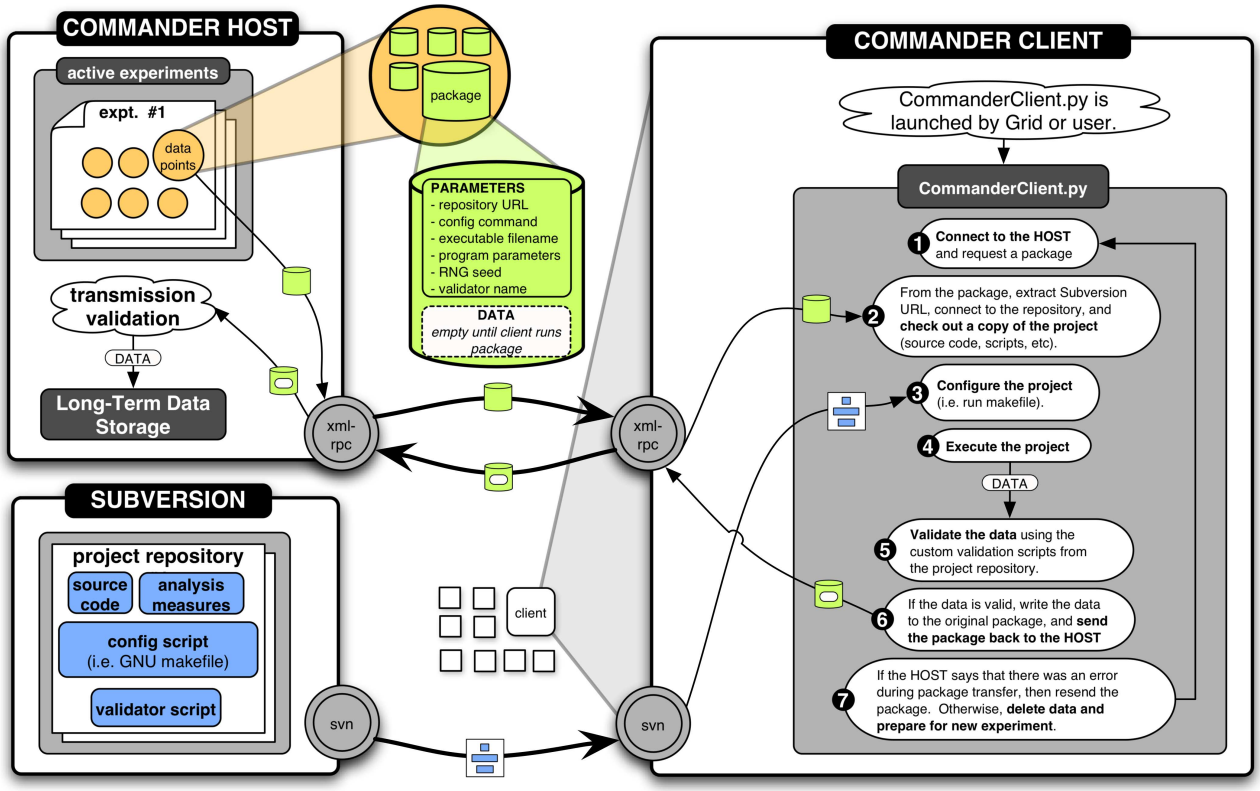


**Figure 1. Commander architectural and processes diagram showing Host, Client, and Subversion server.**

amounts of information immediately available after a GP experiment. This gives the user invaluable feedback about the correctness of the results.

In each of these three stages, Commander separates the tedious tasks from those that require an experimentalist's input. Running an experiment *should* be as simple and transparent as selecting parameter-configurations, selecting the format of results, and waiting for automated data reduction. Although Commander is not novel in either its ability to perform large parameter-sweep experiments or its ability to run processes on remote networks, its automated PSE administration with transparent, robust, distributed computation is unique. As such, Commander's design and architecture is different from previous works.

Unlike previous solutions, Commander relies on no prescribed Grid technology for distributing jobs. Instead, it operates with a *host* and numerous *clients*, relying on any one of numerous Grid protocols or cluster scripts merely to remotely launch the client utility. Commander uses a master-worker architecture in which the host maintains total knowledge of the experiment and replies to clients' requests by sending experiment-specific instructions. To achieve platform-independence, clients are written in Python—we regularly run clients on Unix, Linux, and OSX platforms, simultaneously spread across different Grid architectures, cluster networks, and individual workstations.

When clients request a job from the host, the host returns a *package* with information describing how clients should obtain, compile, and execute the necessary project source code with a given set of input parameters. Commander asserts that all project-specific materials, such as source code, configuration/compilation scripts, and data validation scripts must be kept in a version control system to which clients can connect and download necessary items.

After the client runs a trial, optional validation scripts check data integrity. If the data passes the tests, it is returned to the host for analysis and archiving. Clients continue the cycle of *check out, configure, execute, test, check in* until no more packages remain. When the host determines that all job packages have been successfully evaluated, it applies the user-selected analysis functions to perform automated data reduction. 2D and 3D graphs are automatically generated with independent axes corresponding to swept parameters and the dependent axes corresponding to analysis functions. All trial data is archived in a long-term storage location, where it is available for later more-detailed analysis.

Although parameter sweeps are difficult to conduct, they are useful for experimentalists who want to see a more complete view of GP dynamics. *Commander is our solution to a generic parameter sweep engine for experimentalists*. Commander is designed to automate as much of the experimentation process as possible by alleviating tedious tasks, providing robust remote trial distribution, and ensuring validity in data collection. Through simple interfaces, Commander allows researchers to quickly define new experiments, run them, and use included analysis functions to see the results. This low turnaround time means that experimentalists can be more productive with their experiments. Data can be interpreted, questions can be asked, and science can be achieved in a more productive and responsive fashion. It is our hope that software packages like Commander will encourage the GP community to conduct these valuable experiments more frequently. Commander can be found online at http://www.lattice.umich.edu/commander