Chapter 7
Parametric Analysis with OpenStudio

7.1 Introduction to Parametric Analysis

The previous chapter introduced the concept of OpenStudio Measures and how they can be applied individually and in combination to a Model to create and compare different Design Alternatives. While an improvement from modifying models by hand, generating results, and comparing them; the manual analysis workflow is still labor intensive, non-scalable, and will not necessarily yield the best solution for a given problem. In this chapter, we will discuss how OpenStudio enables automated creation and search of large building parameter spaces. We'll also look at how these same approaches may be used to "tune" models of existing buildings to best match measured energy consumption data.

7.2 OpenStudio Server

In the previous chapter, we alluded to a "server" that ran on a user's computer to manage multiple simulations in PAT's manual mode. To understand and utilize OpenStudio for parametric analysis, it is important to have a better understanding of what OpenStudio Server is and (to a limited degree) how it works. Figure 7.1 illustrates the basic relationship between PAT, the "OpenStudio Server," and the workers who create and simulate the individual data points.

OpenStudio Server has three primary components:

1. **Web Interface** – A minimal interface that allows for user interaction with analysis projects and the contents of the Results Database,

---

The original version of this chapter was revised. A correction to this chapter can be found at https://doi.org/10.1007/978-3-319-77809-9_10

Electronic Supplementary Material: The online version of this chapter (https://doi.org/10.1007/978-3-319-77809-9_7) contains supplementary material, which is available to authorized users.

L. Brackney et al., *Building Energy Modeling with OpenStudio*,
https://doi.org/10.1007/978-3-319-77809-9_7

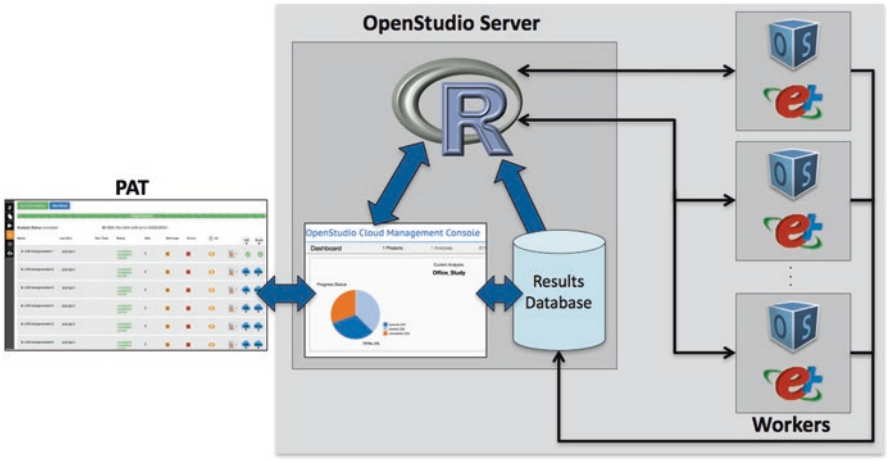

Fig. 7.1 PAT, OpenStudio Server, and Workers

2. **Results Database** – A database used to store high level simulation results for a project, and
3. **R** – An open source platform for statistical computing and analysis.[1]

OpenStudio Workers are a fourth component in the overall architecture, separate from but integral to the server. Each worker node is an independent computing instance, configured with OpenStudio, EnergyPlus, and supporting software. When provided with a seed Model, weather file, and one or more Measures; a worker has everything it needs to create, simulate, and post-process results for a given Design Alternative. While workers generate all of the output available in a typical simulation run, file size and storage capacity generally dictate that only a subset of that data be returned to the Results Database for subsequent use. We'll discuss shortly how Reporting Measures, along with PAT's Outputs ( ) Tab, are used to specify which values are stored.

R does most of the "heavy lifting" for any OpenStudio Server-based analysis, defining individual data points to be simulated. The fundamental difference between the "mini server" used locally for manual analyses and a full OpenStudio Server implementation used in algorithmic mode is the inclusion of R and supporting files. The manner in which R defines data points, reviews results, prescribes additional points, etc. varies based on the algorithm chosen, and is discussed in the following section.

Lastly, OpenStudio Server manages an API that translates PAT projects into problem formulations for R, communication with worker nodes, queuing of simulation jobs, and communication of results back to PAT. Additional details of OpenStudio Server are presented in Chap. 9.

[1] https://www.r-project.org/.

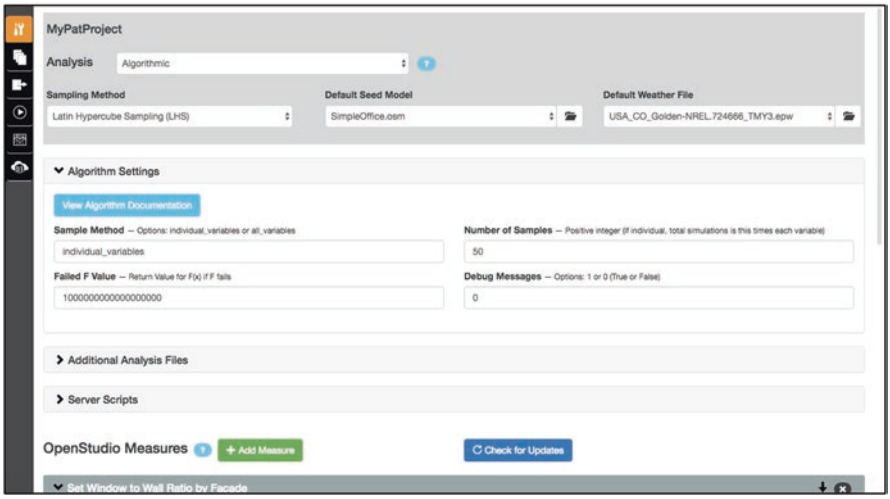

Fig. 7.2 PAT analysis Tab in algorithmic mode

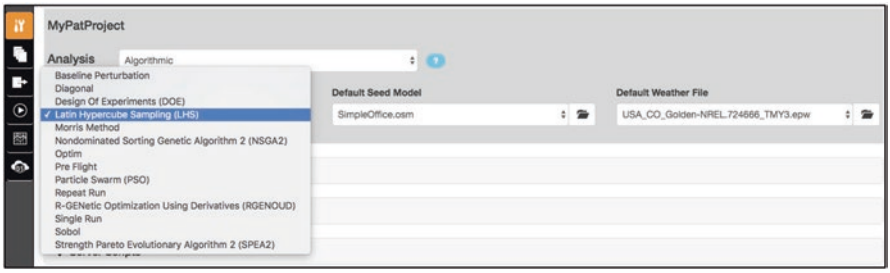

Fig. 7.3 Selecting an algorithm in PAT

7.3 Algorithms in PAT

PAT has been designed to enable large-scale exploration of design spaces using a range of sampling, optimization, and machine learning algorithms. Switching from "Manual" to "Algorithmic" in the Analysis selection field of the Analysis ( ) Tab modifies PAT's interface and functionality in a number of ways. Immediately obvious are the additions of "sampling method" as a selection field near the top of the window along with new sections in the Tab for algorithm settings, analysis files, and server scripts (Fig. 7.2).

In this example, Latin Hypercube Sampling (LHS) is selected as the algorithm that will be used to guide exploration of the design space. Immediately below the sampling method field are collapsible sections denoted by a › Button for algorithm settings, supplementary analysis files, and server scripts. A View Algorithm Documentation Button provides a link to detailed documentation for each algorithm and its settings.

Figure 7.3 shows the contents of the sampling method choice list and the range of algorithms that OpenStudio server currently supports. While complete docu-

mentation for most algorithms is available via View Algorithm Documentation, it's important to have a general understanding of what each algorithm attempts to do and the kinds of applications that each is best suited for.

7.3.1 Single Run

Selecting the Single Run algorithm creates a single data point including the seed Model and each Measure applied with its static/default value. This algorithm is primarily used for testing the seed Model and Measures on a server prior to running a larger analysis.

> **Tip:** This algorithm is so-named because it is intended as a "pre-flight" check of the seed model and measures prior to performing any larger analysis. Pre-flight checks are strongly recommended prior to committing significant computing resources for sampling and optimization problems.

7.3.2 Pre-Flight

"Pre-Flight" takes the Single Run algorithm a step further, creating three data points utilizing the Seed Model with all measures applied using their minimum, mean, and maximum arguments.

7.3.3 Repeat Run

This algorithm replicates the data point defined by the Single Run algorithm a specified number of times. It is primarily used for software development and testing of the OpenStudio Server itself, and is not of general interest.

7.3.4 Baseline Perturbation

This is an experimental algorithm and is not recommended for general use at the time of writing.

7.3.5 Diagonal

This is an experimental algorithm and is not recommended for general use at the time of writing.

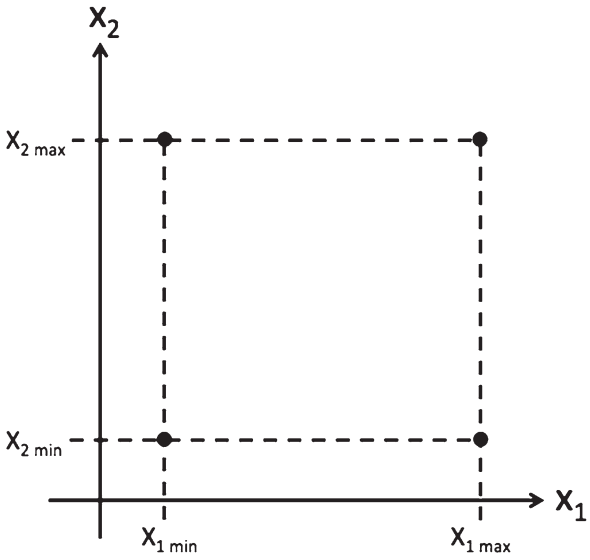

Fig. 7.4 A DOE with two independent variables and two sampling levels

7.3.6 Design of Experiments (DOE)

Design Of Experiments or "DOE" is a sampling method that varies each independent variable separately. The number of samples (sometimes referred to as levels) in a DOE problem determines how many times the effect of each independent variable will be evaluated. So a two level DOE with two independent variables would sample the design space shown in Fig. 7.4 resulting in four total data points.

The "full factorial" DOE implemented in OpenStudio Server generates a total of m^n data points for a problem with "n" independent variables and "m" levels,. With problems of any significant complexity, the total number of simulations to be performed in a DOE quickly grows out of control. For this reason, mathematicians have devised alternate sampling methods that attempt to reduce the number of samples without greatly compromising the quality of an analysis.

7.3.7 Latin Hypercube Sampling (LHS)

Latin Hypercube Sampling,[2] or "LHS," attempts to generate a sparser set of samples than DOE using pseudo-random distributions of independent variables. In LHS, independent variable ranges are divided into segments containing equally probable

[2] McKay et al. (1979).

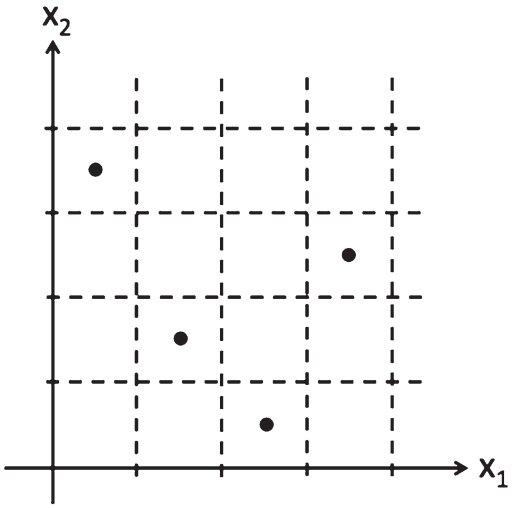

Fig. 7.5 LHS concept with two independent variables and four samples

intervals. The concept of a "Latin Square," a square array in which array elements appear exactly once in each row and column, is applied to those independent ranges to define samples. A simple example involving two independent variables and four samples is shown in Fig. 7.5.

Note that number of samples dictates the number of intervals applied to each independent variable and consequently, the total number of data points that will be defined. For a fixed number of samples, as the number of independent variables increases, the multi-dimensional "grid" discretizing the sample space becomes coarser and coarser, resulting in fewer data points in each dimension, and a rougher approximation of the variable's distribution. We will revisit the concept of independent variable distributions in Sect. 7.4, but for the moment think of LHS as a reasonably efficient means of assessing how different design parameters (Measure arguments) affect building performance.

7.3.8 Morris Method

The previous algorithms generate data points irrespective of their relative performance in terms of energy use intensity (EUI), cost, or whatever other combination of criteria might be of interest. The Morris Method and optimization algorithms described in Sects. 7.3.9 through 7.3.13 all rely on knowing something about how well any given data point achieves a performance objective. Abstractly, we define a performance objective function "F" in terms of our independent variables, the vector **x**. We will discuss exactly how we define F(**x**) using PAT in Sect. 7.5.1, but for the moment imagine that algorithms have a means of quantifying how well any given data point meets an objective like EUI.

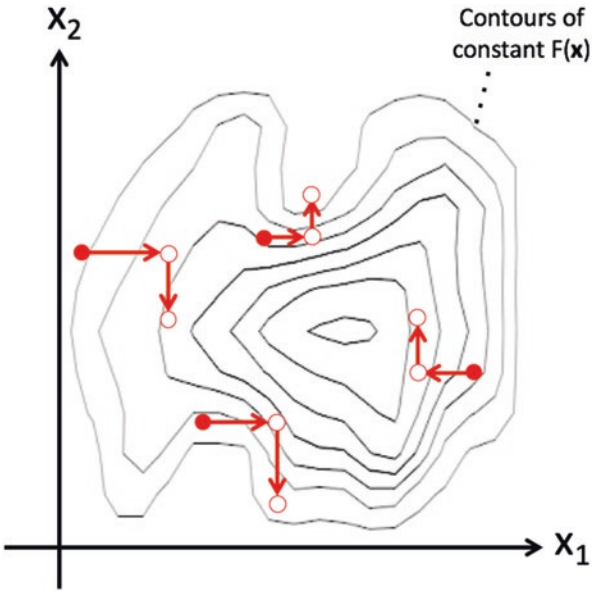

Fig. 7.6 Morris method concept with initial and subsequent data points for r = 4

The Morris Method[3] is used to assess how sensitive the objective function is to changes in the independent variables. For example if window-to-wall ratio and insulation R value were both available as Measure arguments in a PAT analysis, the algorithm can be used to inform which of those variables had a more significant impact on EUI for a particular building. Unlike DOE or LHS, which identify all samples (and associated input argument values) a priori, Morris method starts by "randomly" selecting a set of samples within the available input ranges, simulating those data points, and evaluating the objective function for each. Input arguments are modified, one at a time, creating new data points, simulations, and objective functions. A new starting point is randomly selected and the entire process is repeated a number of times. If r is the number of times and n is the number of independent variables, this produces a total of r * (n + 1) data points (Fig. 7.6).

7.3.9 Sobol Method

The Sobol[4] Method is an alternative to Morris for identifying parametric sensitivity for a problem. Sobol decomposes the variance of outputs into fractions that are attributed to inputs. Sobol typically requires more point evaluations than Morris, providing better coverage of the parameter space at the expense of computation time.

[3] Morris (1991).

[4] Sobol (2001).

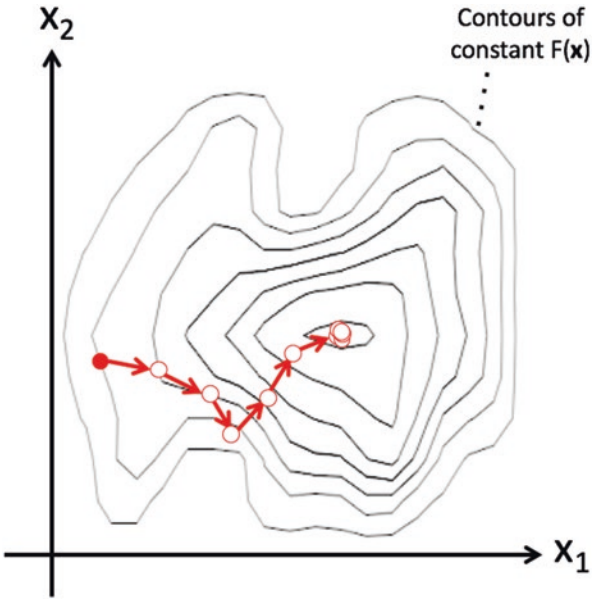

Fig. 7.7 Optim conceptual progression to optimal solution

7.3.10 Fourier Amplitude Sensitivity Test (FAST99) Method

The Fourier Amplitude Sensitivity Test (FAST99)[5] is a third method of estimating parametric sensitivity based on a variation of the Sobol Method. It improves (computationally) on Sobol, and generally falls somewhere between Morris and Sobol in terms of computational performance and accuracy.

7.3.11 Optim

Optim is the simplest goal-seeking algorithm supported by OpenStudio Server. It is based largely on a Nelder-Mead Simplex method[6] that moves in single directions (linear combinations of **x**) until no further improvement in F(**x**) is achieved. Figure 7.7 shows a conceptual progression of a Simplex approach for two independent variables and a well-behaved performance surface. Because optim calculates approximate derivatives for the performance index, it is not recommended to apply it to problems with non-differentiable surfaces. This includes design problems with discrete independent variables (e.g. wall insulation values of R20, R30, R40, etc.)

[5] Saltelli et al. (1999).

[6] Nelder and Mead (1965).

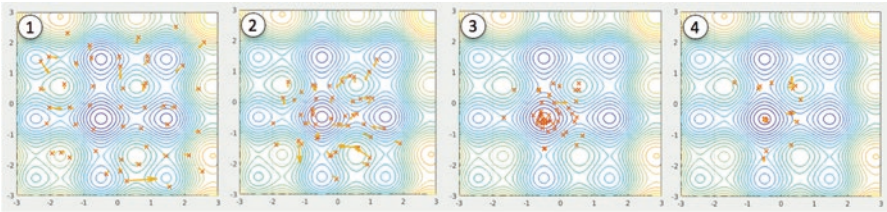

Fig. 7.8 Example particle swarm optimization progression (Figure adapted from Ephramac, 2017)

Since a great many building optimization problems of interest fall into this category, optim should always be used with an abundance of caution.

7.3.12 Particle Swarm Optimization (PSO)

Particle Swarm Optimization was originally intended to simulate social behavior, and loosely modeled the behavior of flocks of birds and schools of fish.[7] In this algorithm a set of particles (data points) migrate around the design space based on an equation that governs their relative velocities, proximity to local optimae, and proximity to the most optimal solution observed by the swarm. Over time, the swarm generally drifts towards the best known solution, but individual particles are granted a degree of autonomy to explore local optimae that may yield better solutions (Fig. 7.8). This potentially allows the algorithm to identify a global optimum when performance surfaces contain multiple local minimae. Because the algorithm does not require the performance surface be differentiable, it is better suited to address a broader class of optimization problems than "optim."

7.3.13 Nondominated Sorting Genetic Algorithm 2 (NSGA2)

The last three optimizers discussed in this section are based (at least in part) around a class of solvers referred to as "genetic algorithms," which attempt to digitally mimic the concept of natural selection.[8,9] In a genetic algorithm, the independent variables are typically discretized or "encoded" to construct a population of solutions. Figure 7.9 contains examples for three data points with roof and wall insulation, lighting, and window-wall-ratio as independent variables; effectively identifying the "DNA" of a particular building. In the case of variables that are already discrete (e.g. insulation values of R20, R30, R40, etc.) no encoding is necessary – this enables genetic methods to address building design problems that other algorithms cannot.

[7] Kennedy and Eberhart (1995).

[8] Barricelli (1957).

[9] Fraser (1957).

Design A	R30 Roof	R30 Wall	T8 Lights	20% WWR
Design B	R30 Roof	R30 Wall	T12 Lights	20% WWR
Design C	R40 Roof	R40 Wall	T12 Lights	40% WWR

Fig. 7.9 Genetic algorithm "DNA" for three building data points

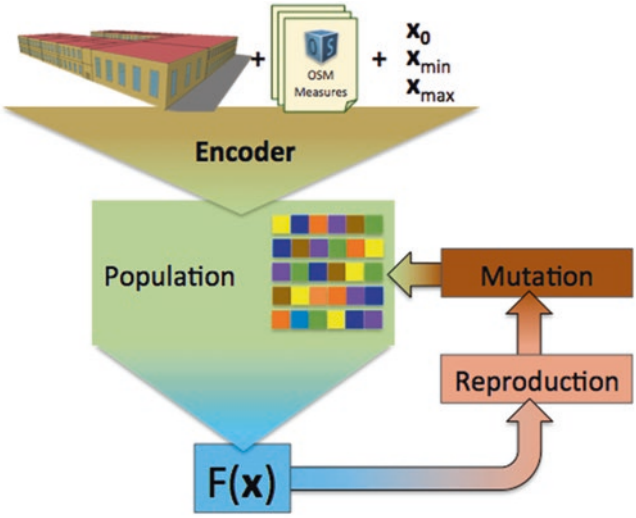

Fig. 7.10 Genetic algorithm applied to a building optimization problem

Once a population has been established, the data points are simulated and F(**x**) is evaluated. The next generation's "offspring" are created by combining data points (parents) that performed well. Poorly performing data points are removed from the "gene pool," and the process repeats. Genetic algorithms may also introduce the concept of "mutation," random modifications to an offspring's DNA that enable the algorithm to identify better solutions by chance. This iterative process is illustrated in Fig. 7.10.

Unlike other optimizers, NSGA2 does not require a scalar objective function. As generations progress from an initially random population, a locus of points, **F**(**x**), is generated that represent the best identified solutions that have been observed. This locus is called a "Pareto front," and represents a collection of

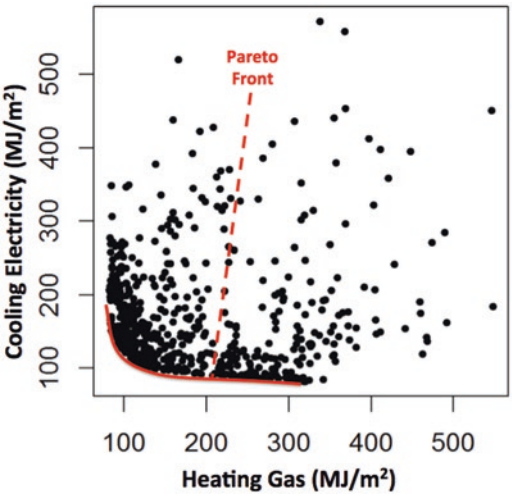

Fig. 7.11 Multi-variate optimization of heating and cooling energy use with Pareto Front

solutions that are "good" in some sense. Figure 7.11 illustrates the results of a multi-variate optimization that seeks to minimize heating and cooling energy use. Which solution is best? The answer to that question is – it depends. For this problem do we value cooling electricity use more than heating gas consumption? The Pareto Front illustrates the tradeoff between these competing goals, and the "best" solution likely exists somewhere on that curve. One solution might be to apply the associated energy costs as weighting factors to effectively turn the problem into a univariate optimization problem, but as we'll see in subsequent examples, a singularly optimal solution is rarely that straightforward.

In the case of the NSGA2 algorithm, the relative distances between data points and the Pareto Front are used to sort or rank solutions for pairing and reproduction.[10] The algorithm also gives preference to solutions which are not clumped together near other solutions, promoting their genetic characteristics into subsequent generations. As with other genetic algorithms, the process proceeds until a specific number of generations has been reached, or an optimization convergence criteria has been met.

7.3.14 Genetic Algorithm with Island Evolution (GAISL)

GAISL is another genetic algorithm variant that partitions populations into isolated "islands" for reproduction and mutation.[11] Occasionally a data point will "migrate" to another island, allowing subpopulations to share genetic material.

[10] Deb et al. (2002).

[11] Belding (1995).

7.3.15 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

As its name might suggest, SPEA2 is also a multi-objective optimization algorithm that makes use of Pareto Front information to inform how the algorithm progresses. SPEA2 is also a genetic algorithm and shares many similarities to the NSGA2 approach. SPEA2 differs in how it utilizes Pareto information to cluster data points and match pairs for propogation into subsequent generations.[12] Comparison of SPEA2 and NSGA2 for a specific power system dispatch optimization problem suggests that SPEA2 is capable of finding slightly better/more diverse sets of solutions than NSGA2 at the expense of increased computation time due to its computational complexity.[13] The authors have not performed extensive comparisons of these two algorithms for building design optimization problems.

7.3.16 R-GENetic Optimization Using Derivatives (RGENOUD)

RGENOUD, is a hybrid solver that combines the strengths of a genetic algorithm (discrete variable capability, robustness in finding global optimum, etc.) with the potential speed of gradient-based solvers like optim.[14] In the case of RGENOUD, derivative estimates inform propogation between generations to quickly climb hills or decend valleys of the performance surface without significantly compromising the algorithms stability and likelyhood of identifying a global minima. The authors regularly use RGENOUD for building optimization problems, and generally that it offers a reasonable trade-off between computation speed and performance.

7.4 Working with Measure Variables and Arguments

Algorithmic mode significantly alters the Measure section of the Analysis ( ) Tab. The concept of manual mode's Measure "option" no longer applies, and variable checkboxes now become selection fields that allow the user to specify the nature of Measure inputs. Measure inputs may be set to one of four types in PAT:

1. **Argument** – Fixes the quantity as a static value for the analysis.
2. **Continuous** – Assigns a continuous probability density function to a variable.
3. **Discrete** – Allows the user to specify a specific set of distinct values with associated weights.
4. **Pivot** – Forces the entire analysis to be repeated against each specified (discrete) value.

[12] Zitzler and Thiele (1998).

[13] King et al. (2010).

[14] Mebane and Sekhon (2011).

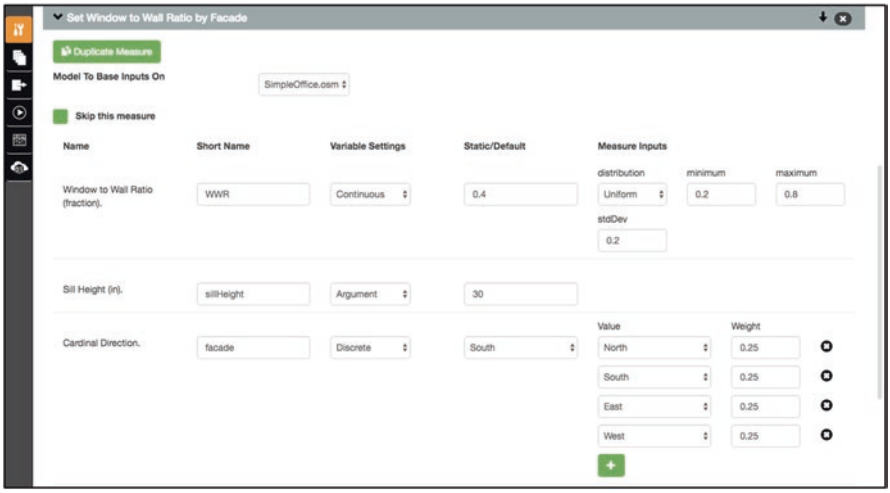

Fig. 7.12 Assigning multiple variable types to a Measure's inputs

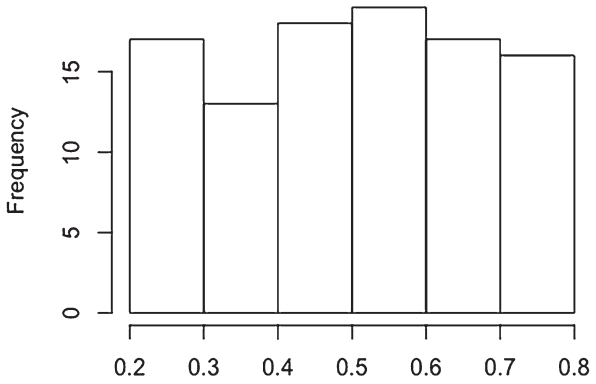

Fig. 7.13 Uniform distribution for window to wall ratio

By default, all Measure inputs are considered arguments unless explicitly changed to another type. This allows the user to vary, for example, only the window-to-wall ratio in the Set Window to Wall Ratio by Façade Measure. Multiple inputs in a single Measure may vary, and they need not all be defined as the same variable type. Figure 7.12 provides a good example involving the Window to Wall Ratio by Façade Measure. In this case, the window to wall ratio is a continuous variable, sill height is a constant, and the cardinal direction of the windows changed by the Measure will vary according to a discrete distribution.

Figure 7.13 shows the distribution of window to wall ratio values that the Measure might apply to the seed Model over 100 simulations. As the number of simulations increases, the histogram would become flatter, better approximating the uniform distribution requested.

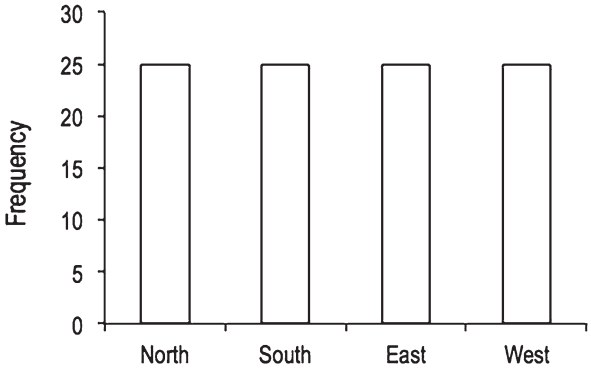

Fig. 7.14 Discrete distribution for cardinal direction

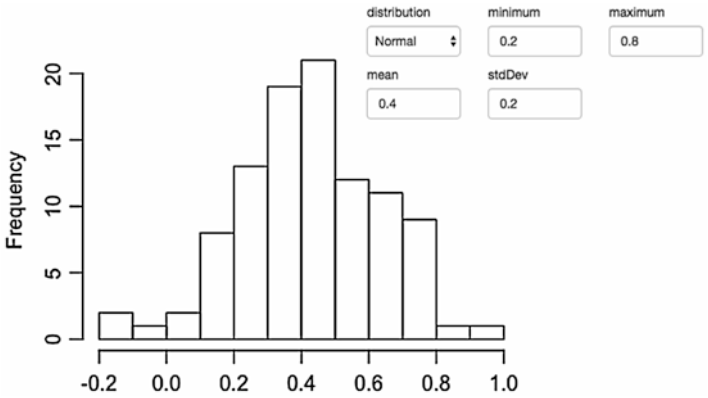

Fig. 7.15 Normal distribution for window to wall ratio

A discrete distribution for the side of the building the Measure will alter windows on is shown in Fig. 7.14 for a population of 100 data points. The discrete option is appropriate here because the cardinal direction argument in the Measure is a choice list comprised of North, South, East, and West. Additional options may be added by clicking the + Button and removed with the ∘ Buttons next to each value/weight pair. The weight values are normalized to determine how frequently a value might appear in a population. In this case, each of the options has a 25% chance of occurring.

PAT supports three additional types of distributions for sampling problems involving continuous variables: normal, log normal, and triangular that are commonly used in statistical analysis and modeling. Most readers will likely be familiar with the normal distribution shown in Fig. 7.15. Changing the distribution type in PAT adds a mean input field. In this case, we've requested that the algorithm produce a distribution of window to wall ratio values with a mean of 0.4 and standard deviation of 0.2. The histogram shows a possible distribution of Measure inputs over 100 simulations.

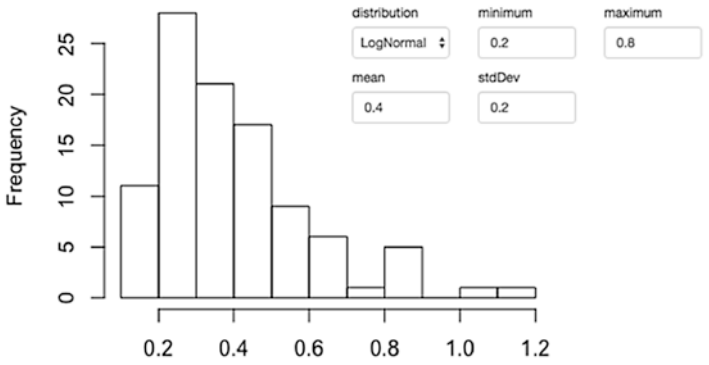

Fig. 7.16 LogNormal distribution for window to wall ratio

Figure 7.16 illustrates distribution of window to wall ratios one might expect when a log normal distribution is applied to a population of 100 simulations.

Note in both the normal and log normal cases that the minimum and maximum entries in PAT do not limit the possible input values that might be generated by a sampling algorithm.[15] In the case of the normal distribution there are extreme results in the distribution's tail with negative window to wall ratios. In the log normal case, the Measure would be asked to modify the seed Model to produce window to wall ratios exceeding 1.0, also a physical impossibility. Well-written Measures will "trap" such situations and limit inputs to reasonable values, reporting a warning in the process. We'll discuss best practices for writing Measures in Chap. 9, but is it safe to assume that all Measures will be sufficiently robust against nonsensical inputs? Because of this possibility, the authors recommend using the triangular distribution in lieu of others.

Figure 7.17 illustrates a potential triangular distribution of window to wall ratio over 100 samples. Results are strictly constrained by the minimum and maximum values, with the peak of the triangle occurring near the mean (or more accurately the mode) of the triangular distribution. Triangular distributions can be used to roughly approximate both normal and log normal distributions, while ensuring that Measure inputs and resulting data points are reasonable.

That final variable type supported by PAT is a "pivot." At first glance, Pivots might appear to behave like discrete variables, as the user can enter values with the + Button and remove them with the ∘ Buttons. However, one complete sampling problem is performed for each value of a pivot variable. An analysis problem with 100 samples and 3 pivots would create a total of 300 data points to be simulated. Why might you use a pivot variable? One common use for them is not in performing an analysis of a single building, rather studying how efficiency measures might apply to many types of buildings. Figure 7.18 illustrates just this situation.

[15] While samplingalgorithms (e.g. LHS) will not respect prescribed maximum and minimum values, they are used to constrain solutions generated by PAT's optimizers.

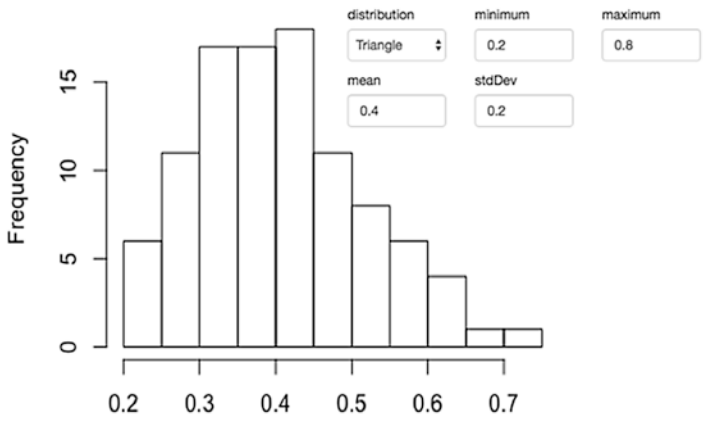

Fig. 7.17 Triangle distribution for window to wall ratio

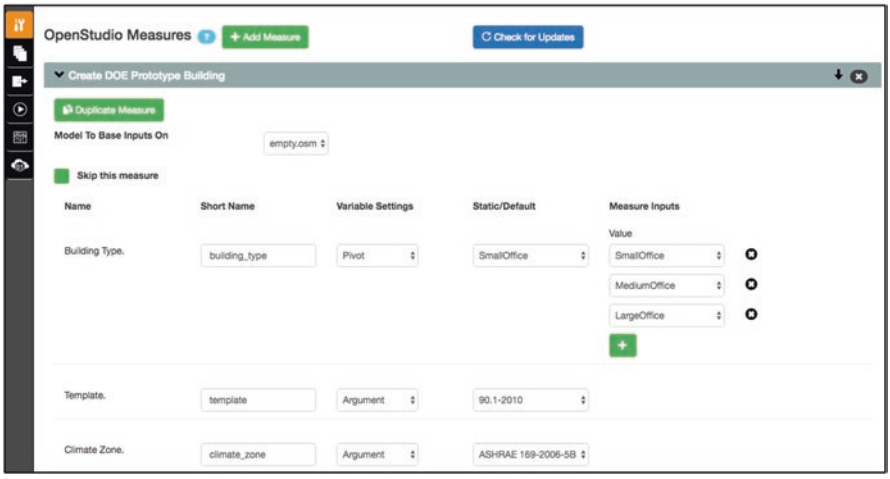

Fig. 7.18 "Pivoting" an analysis on building type

In this example, the "Create DOE Prototype Building" Measure and OpenStudio Standards Gem introduced in Chap. 6 are used to create an entire building Model from scratch. Note that the seed Model in this project is called "empty.osm," which is literally an empty Model. The building type input for the Measure is set as a pivot with Small, Medium, and Large Office as values. Any of the Measure's 13 other building types including schools, hospitals, hotels, and retail could also be added to expand the study. The building template and climate zone inputs are left as arguments for ASHRAE 90.1-2010 and climate zone 5B respectively. Combining this with the "Set Window to Wall Ratio by Façade" Measure along with an appropriate sampling algorithm allows us to investigate the effect of window variation on multiple building types. This is an excellent example of the kind of flexibility that OpenStudio, Measures, and PAT provide.

7.5 Other Changes to PAT in Algorithmic Mode

In addition to the option to select and configure algorithms in the Analysis ( ) Tab, a number of other changes manifest throughout PAT's tabs – some obvious and some subtle. For example, since an algorithm is used to specify all Design Alternatives, there is no use for PAT's Design Alternatives ( ) Tab as shown in Fig. 7.19.

7.5.1 The Outputs Tab in Algorithmic Mode

Analysis via algorithm tends to produce very large data sets. As such, the methods for visualizing results and teasing out valuable insights differ from the simple table views available in manual mode. In addition, some algorithms (e.g. optimizers) are goal seeking, and require specification of performance metrics to function properly. PAT's Outputs ( ) Tab, shown in Fig. 7.20, is designed to specify key simulation outputs for use in post-processing large sets of simulation results or as a means of guiding optimizers. At a minimum, the Outputs ( ) Tab needs the OpenStudio Results Measure to be available to provide a nominal set of outputs, although additional reporting Measures may be used to add more outputs.

Pressing the Select Outputs Button brings up a dialog of all outputs specified by the reporting Measure (Fig. 7.21). The user may select all of the outputs, or only individual outputs via check boxes.

Fig. 7.19 PAT's Design Alternatives Tab in algorithmic mode

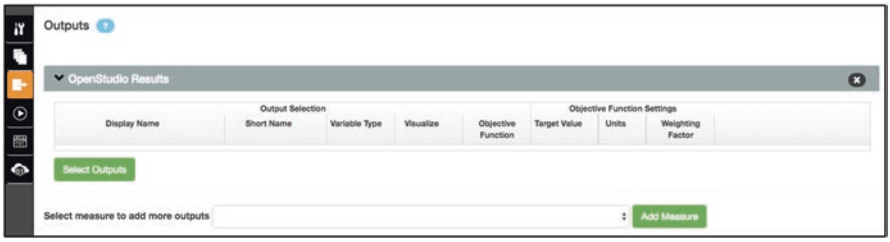

Fig. 7.20 PAT's Outputs Tab in algorithmic mode

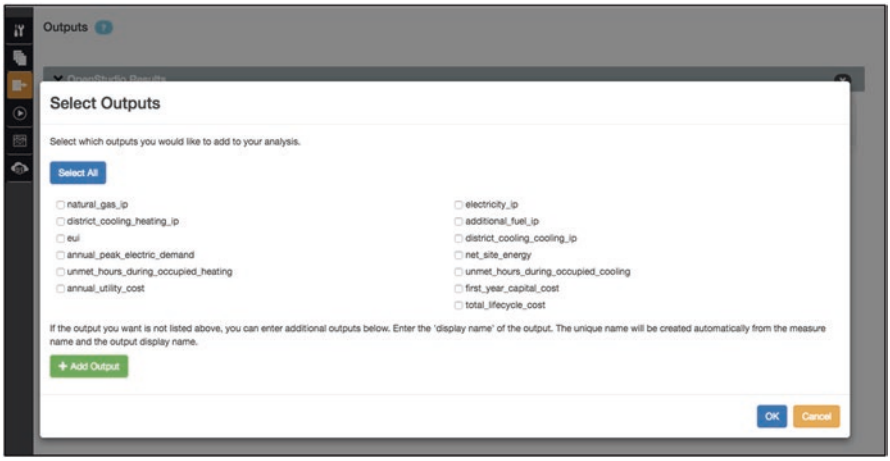

Fig. 7.21 Outputs available from the OpenStudio Results Measure

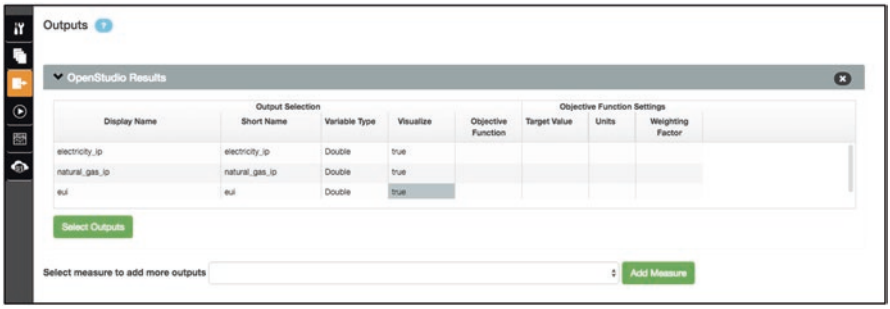

Fig. 7.22 Configuring selected outputs

Once selected, outputs appear in the Outputs ( ) Tab along with additional fields (Fig. 7.22). The "visualize" field should be set to true for any variables the user wishes to preserve from worker node simulations for use in visualizations or subsequent analysis.

The remaining columns in the Outputs ( ) Tab are used to specify objective function, F(**x**). They include:

1. **Objective Function** – Should be set to true for any variables that will be included as part of an objective function (an element in vector **x**).
2. **Target Value** – The target value for the output variable that will be sought by the goal-seeking algorithm.
3. **Weighting Factor** – A multiplier used to create linear combinations of error functions in multi-objective optimization problems.
4. **Objective Function Groups** – This argument only appears when an algorithm that performs multi-objective analysis is in use. Use an integer to group performance indices or assigning each index to its own group number.

It is important to note that many optimization algorithms require scalar performance metrics. Multi-objective optimization is frequently achieved through linear combination (weighted average) of separate performance indices as shown in the following equation:

$$F(\boldsymbol{x}) = w_1\sqrt{(f_1 - t_1)^2} + w_2\sqrt{(f_2 - t_2)^2} + w_n\sqrt{(f_n - t_n)^2}$$

where:

w_i is the *i*th weighting factor,
f_i is a performance metric computed by simulating a data point, which is a function of **x**, and
t_i is a target value associated with the *i*th performance metric.

The $\sqrt{(f_i - t_i)^2}$ terms in the equation represents an "L^2 norm" of the error between metric and target. Although some algorithms can use other norms, the L2 norm is commonly used by goal-seeking algorithms that require a positive definite performance index.

7.5.2 The Run Tab in Algorithmic Mode

The size of algorithm-based analysis problems generally exceeds the computational capability of a personal computer. PAT has been designed to run these sorts of analyses in the cloud or on dedicated servers. While PAT's "mini server" still launches in the background, the application does not support algorithms using local computing resources. For that reason, PAT's Run ( ) Tab will initially look like Fig. 7.23 when first selected in Algorithmic Mode.

Selecting "Run on Cloud" as the Run option changes the Run ( ) Tab significantly (Fig. 7.24).

Run on Cloud produces a dialog that enables the user to specify either an "Existing Remote Server" or Amazon Cloud. Provisioning an existing, dedicated

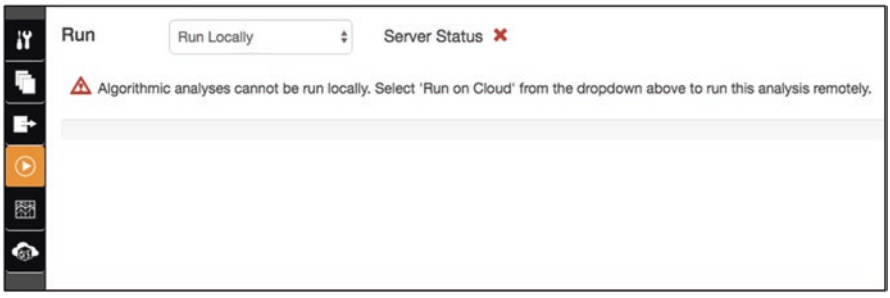

Fig. 7.23 PAT's Run Tab in algorithmic mode

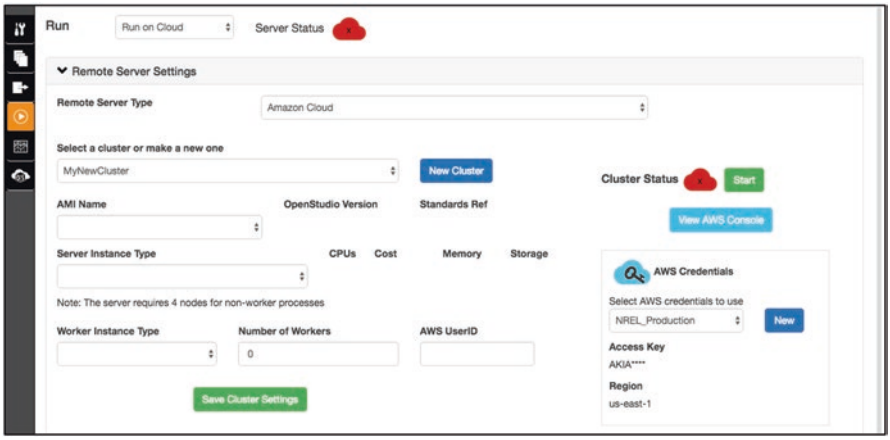

Fig. 7.24 PAT's Run Tab with Run on Cloud selected

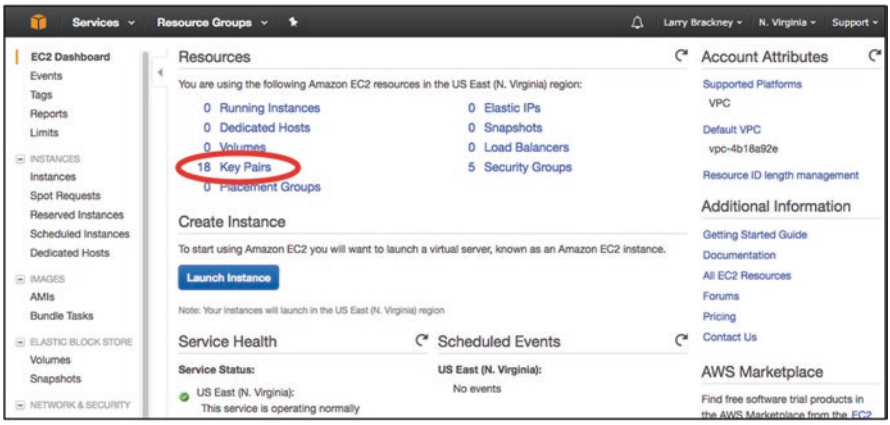

Fig. 7.25 AWS console with secret key option highlighted

server with OpenStudio is beyond the scope of this text. Interested users are referred to https://github.com/NREL/OpenStudio-server for further guidance on the topic.

Amazon Web Service (AWS) has been pre-provisioned with Amazon Machine Images (AMIs) for OpenStudio Server that enables users to leverage large-scale computing with minimal effort and cost. New users will need to establish an account at https://aws.amazon.com. The AWS web console enables tracking and management of computing resources, security credentials, billing details, etc. Figure 7.25 depicts the AWS console with a link to its "Key Pair" management tool highlighted. Establishing a key pair is the first step in linking OpenStudio to AWS. Clicking on this link takes the user to a management screen with a Create Key Pair Button allowing creation of a new set of credentials. Follow the instructions on AWS and make note of the new access and secret keys before proceeding.

The following steps begin the process of setting up PAT to use AWS as a remote computing cluster for large-scale analysis:

1. In the PAT Run ( ) Tab, select "Run on Cloud" with "Amazon Cloud" as the server type.
2. Press the New Cluster Button and choose a cluster name to store your AWS cluster settings.
3. Press the New Button in the AWS Credentials area and follow the prompts to enter the access and secret key information created from the AWS console.

The cluster name is used to save PAT's settings for AWS including the number of workers, size of compute nodes, etc. Multiple setting sets may be stored for use as an analysis is scaled up. Like the cluster name, the AWS credential name is used by PAT to help the user manage access credential configurations, and multiple sets may be stored to assign analyses to different AWS accounts.

A field labeled "AMI Name" specifies the version of OpenStudio that the server and worker nodes will use. This list may include a large number of major and minor OpenStudio versions. It is recommended that you use a major release (one ending in a .0) that corresponds with the version of OpenStudio installed on your computer. The specific version of OpenStudio and the Standards Gem included in the AMI are listed next to the AMI Name field.

The Server and Worker Instance Type fields allow the user to configure the processing capability and storage size of the primary server and workers used to perform an analysis. The user is referred to AWS documentation related to server and worker options, but PAT provides a brief description of the node configuration and approximate cost/hour next to the server choice field. A PAT analysis requires a server and a minimum of one worker. Figure 7.26 illustrates a cluster configured to run an analysis with an "m3.2xlarge" server and 2 m3.2xlarge workers at a total cost of approximately $1.68[16] per hour. Each machine has 8 CPUs so a total of 16 workers along with a few of the server's spare nodes are available for simulation.

Once configured and saved with the Save Cluster Settings Button, the user should press the Start Button next to the Cluster Status indicator on the right side of the UI. A dialog (Fig. 7.27) explaining the user's responsibility to monitor and manage AWS computing resources appears and must be acknowledged before proceeding.

At this point, the server and worker provisioning process begins as shown in Fig. 7.28. A startup dialog states that the process may take many minutes.

Clicking on the View AWS Console Button will open the AWS console in a web browser, allowing the user to monitor the startup process as shown in Figs. 7.29 and 7.30.

Once the server is running, the Server Status indicator will change from to . Two new Buttons also appear in the Run ( ) Tab: Run Entire Workflow and View Server. Clicking the Run Entire Workflow Button starts the analysis on the server. As in Manual mode, progress may be monitored within PAT as shown in Fig. 7.31. The View Server Button functionality will be discussed in the next section.

[16] Amazon routinely updates its pricing structure. Prices listed in PAT are only estimates, and the user is ultimately responsible for knowing what costs may be incurred by an analysis.

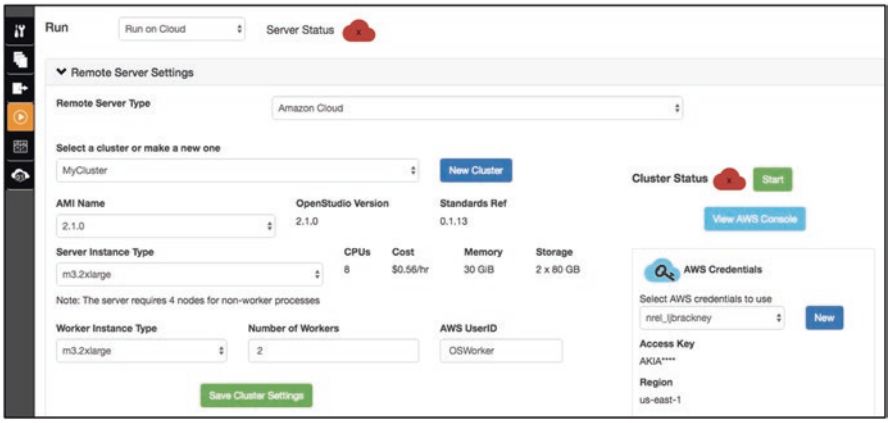

Fig. 7.26 Server and worker node configuration in PAT's Run Tab

Fig. 7.27 PAT AWS responsibility dialog

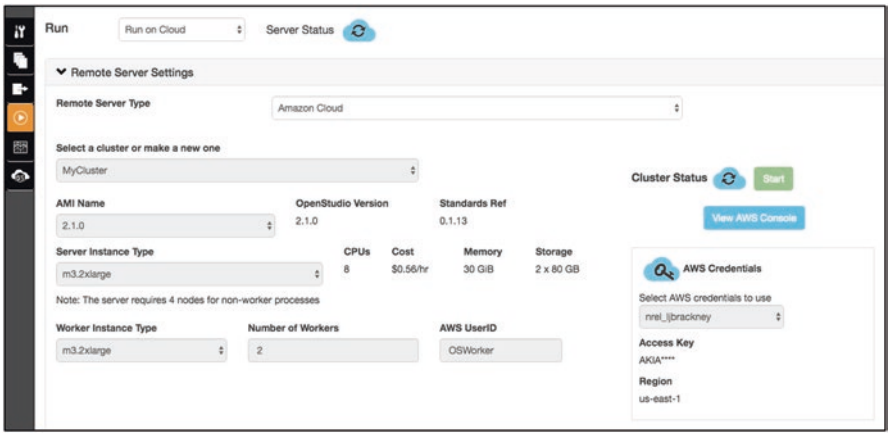

Fig. 7.28 AWS cluster starting up in PAT

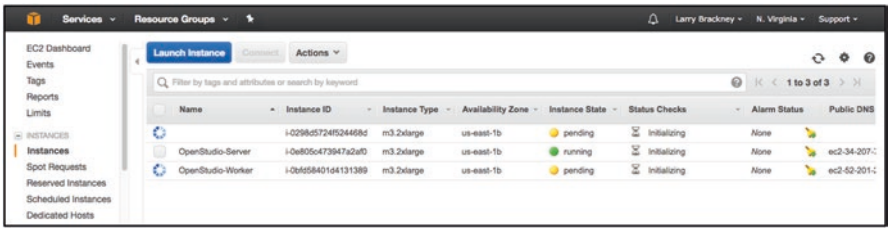

Fig. 7.29 AWS cluster starting up in the AWS console

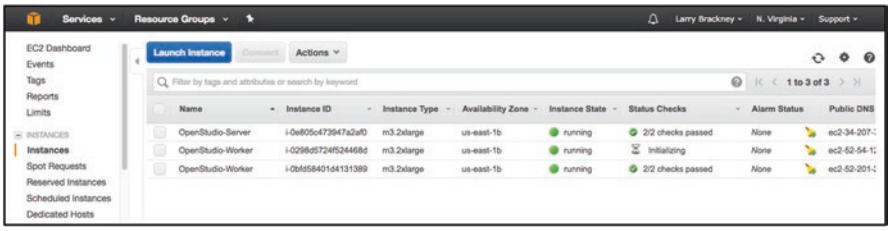

Fig. 7.30 AWS cluster nearly ready for use in the AWS console

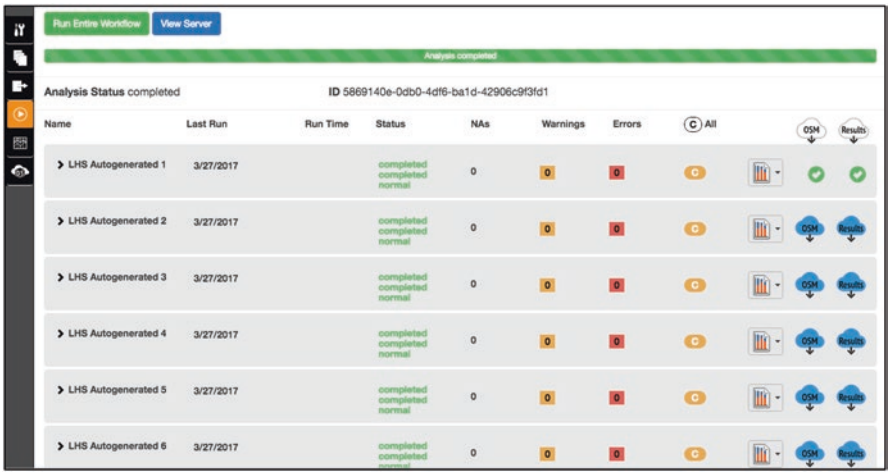

Fig. 7.31 Cloud run completed

One notable difference between running in manual mode with local computing resources and on the server is that detailed simulations are not automatically downloaded from the cloud. These files can be quite large. Clicking on the or Buttons next to a data point will download the OpenStudio Model or completed data point zip file. The cloud Buttons appear as a when a download has occurred. Data point files are lost when the server shuts down, so it is important to download results that may be of particular interest while the server is running.

When you are finished using AWS, it is important to shut the cluster down properly. This can be accomplished using the Terminate Button. Quitting PAT will also bring up a dialog providing the user with the choice of terminating the cluster or leaving it running for the next PAT session to connect to. AWS nodes continue to incur costs to the user regardless of whether simulations are running or not. It is recommended that the user quickly check the AWS console when finishing work to make sure all instances are terminated as expected.

7.6 Working with OpenStudio Server

The OpenStudio Analysis Server ( ) Tab is of much greater importance with algorithmic workflows. This same content may also be accessed through any web browser by clicking the View Server Button in the Run ( ) Tab. The top-level view of OpenStudio server, shown in Fig. 7.32, provides a summary of completed or in-progress projects and analyses along with navigation options.

A single OpenStudio Server analysis (a specific run of a project) is shown in Fig. 7.33. This view provides a high-level summary of the project, links to more detail about the analysis, and status updates for all data points that have been completed, queued for simulation, or are in process. A few important links on this page include:

1. **Project Log** – Near the bottom of the Analysis Information box is a view log file link, which can be helpful in debugging failed analyses.
2. **Downloads** – These links download high-level information and simulation results as CSV or R Data Frames for subsequent analysis. These results include only analysis inputs and outputs that have been defined in PAT. Detailed simulation results associated with individual data points must be downloaded in PAT

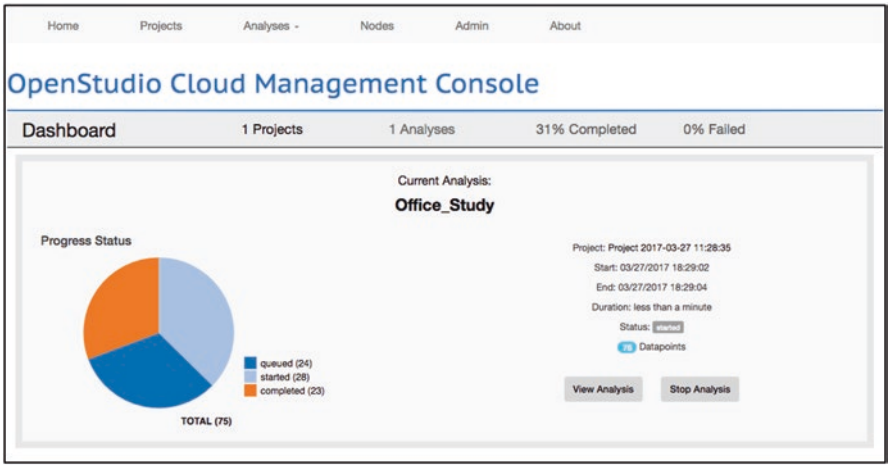

Fig. 7.32 OpenStudio Cloud Management Console front page

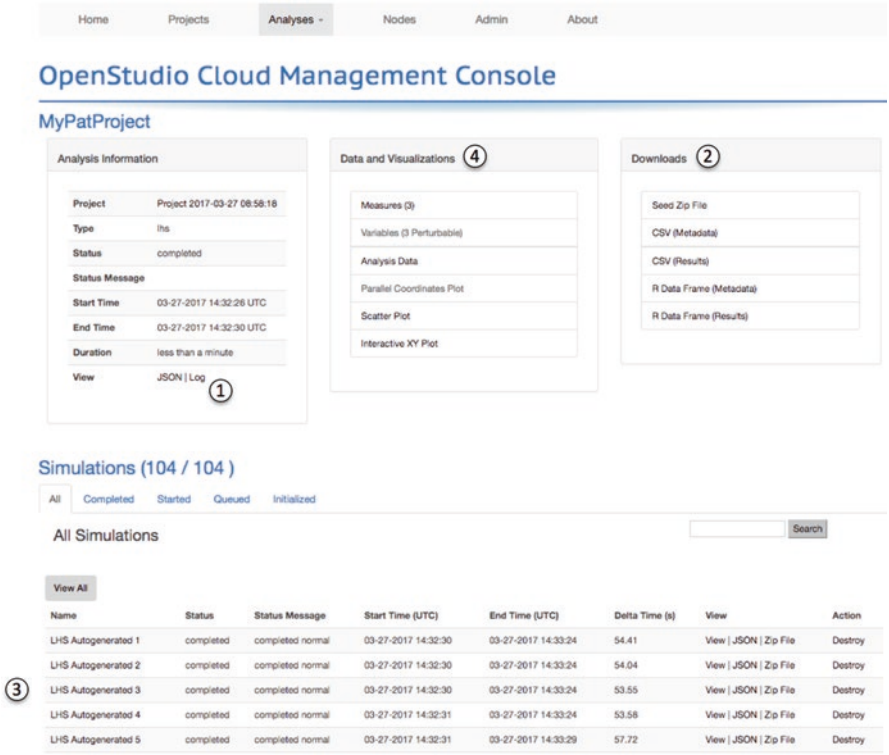

Fig. 7.33 OpenStudio Cloud Management Console view of a project

or via individual data point web links. Additional downloads may be available here upon completion of an analysis depending upon the algorithm being used.

3. **Data Points** – The bottom of the analysis page includes a snapshot of all data points along with their status, run times, and data point-specific links including a zip file containing models and simulation results.
4. **Data and Visualizations** – A number of useful project summaries and interactive visualization tools are built into OpenStudio server.

The variables link near the top center of the analysis web page provides a concise summary of variables and arguments utilized by the analysis algorithm. While any applicable Measures are summarized in an adjacent page, this section provides more detail about how data points have been generated. In the example below, building type is used as a pivot variable as shown in Fig. 7.34. A five level "Design of Experiments" with lighting power density reduction percentage and window to wall ratio variables complete the study space.

Parallel coordinate plots are one of the useful visualization tools built into OpenStudio Server (Fig. 7.35). They provide an interactive means of exploring large data sets and teasing out valuable insights. The plotting tool enables the user to select inputs and outputs that have been pre-defined in PAT's Analysis and

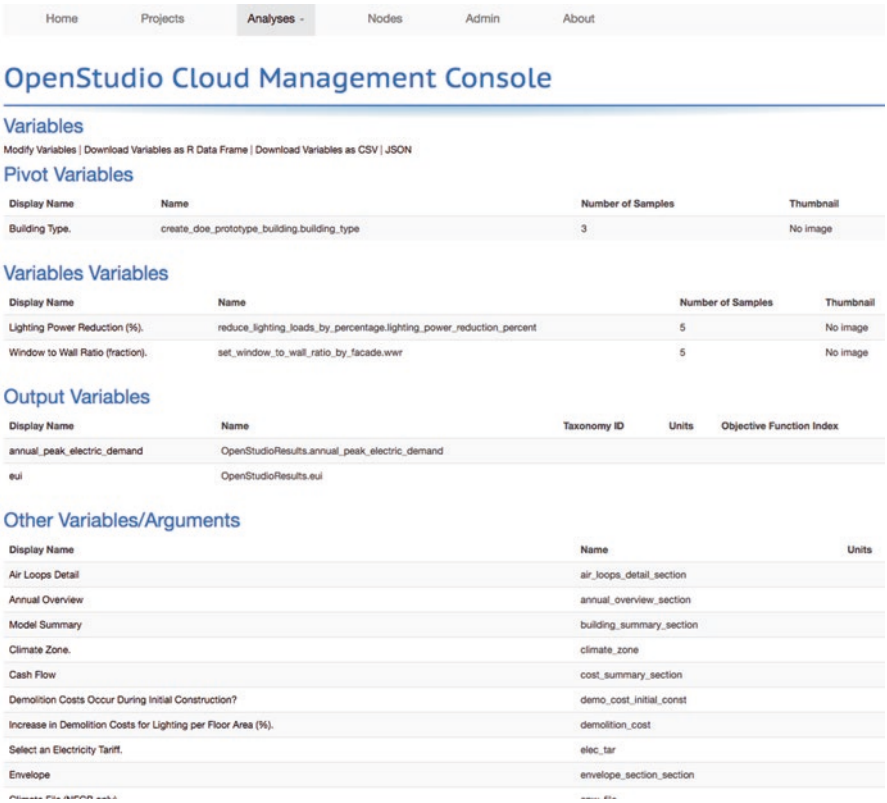

Fig. 7.34 OpenStudio Cloud Management Console project variable summary

Outputs Tabs. Checkboxes turn inputs on and off, and the individual plot axes can be re-ordered via drag and drop. In this example, LHS is used to sample the parameter space for lighting power density, window to wall ratio, and the facade on which windows are placed. Energy Use Intensity (EUI) is selected as the output of interest.

Each blue-segmented line in the parallel coordinate plot represents a single data point. Segments connect key simulation inputs with the resulting outputs. In the above example: lighting power density reduction, window to wall ratio, and window cardinal direction inputs appear on the left side of the chart with EUI as the output of interest on the far right. Figure 7.36 is an annotated parallel coordinate plot highlighting a single data point in a net zero building study performed with PAT.

More than a static display, the user can filter the data points based on input or output ranges to zero in on simulations with certain characteristics. For example, Fig. 7.37 shows how filtering is used in the net zero study to highlight those data points that are net zero or net positive energy producers. We'll make use of this capability in subsequent sections to gain insight into modeled performance.

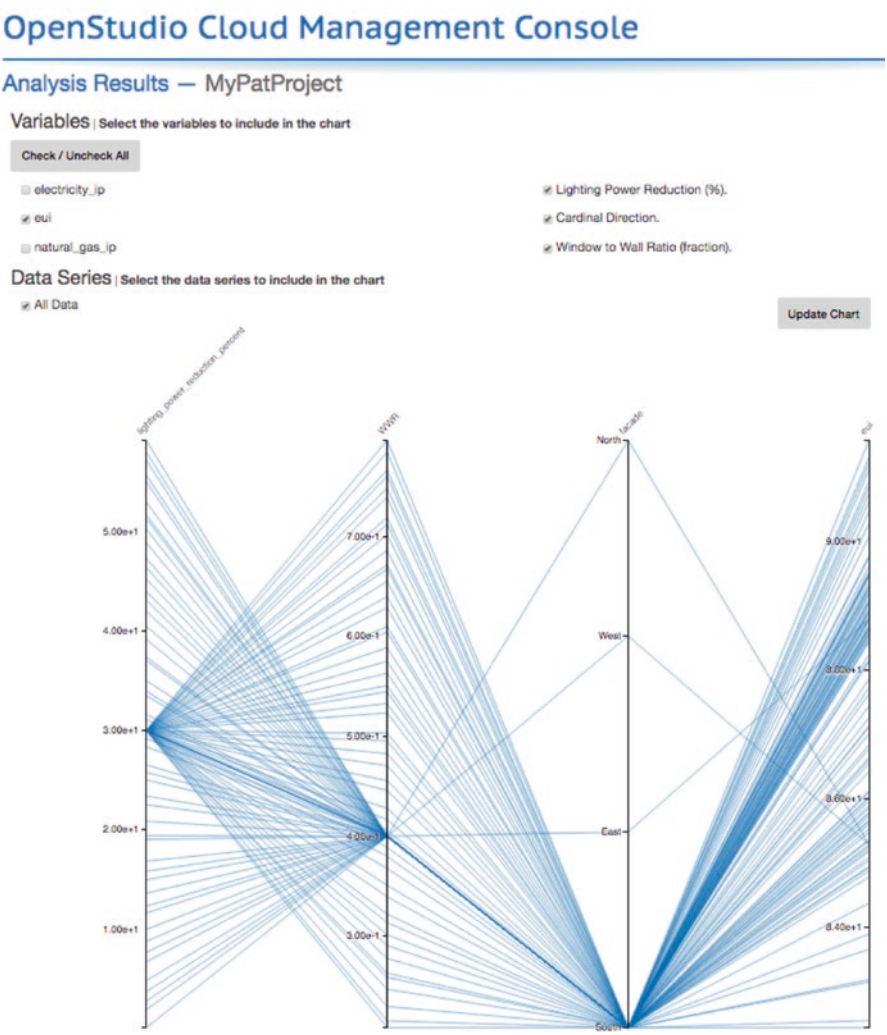

Fig. 7.35 OpenStudio Cloud Management Console parallel coordinate plot visualization

7.7 Checkpoint Ten: Sampling Problems and Uncertainty Analysis

The next two checkpoint exercises assume you have created an AWS account to use EC-2 resources, or that your organization has provisioned a dedicated OpenStudio Server for you to use. The most intensive of these exercises can be analyzed in 10–20 min using 40 or more worker nodes. This first exercise focuses on establishing the analysis setup that we will use in Checkpoint Eleven to solve a problem that building scientists and practitioners are often faced with – Model

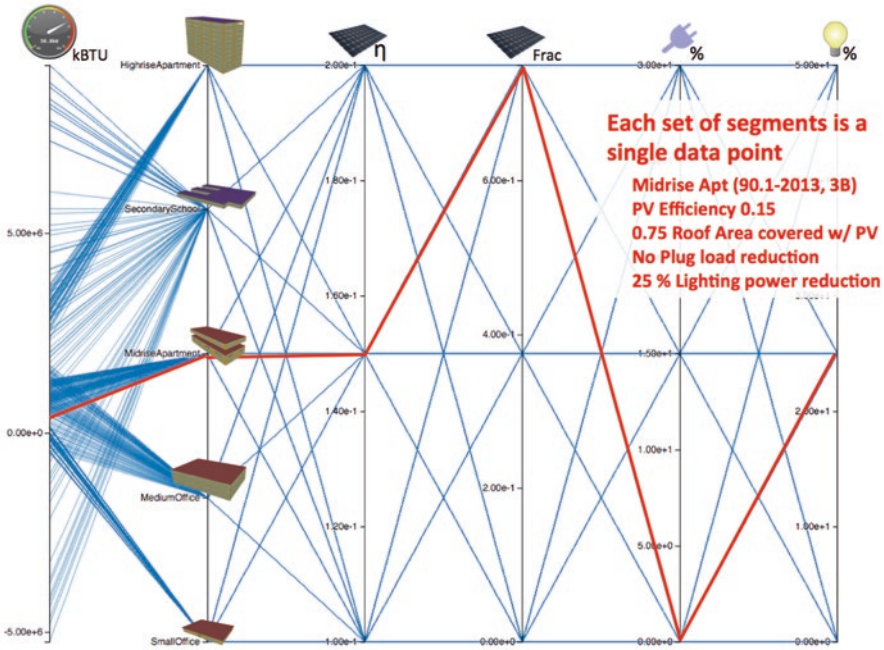

Fig. 7.36 Net zero building study visualized as a parallel coordinate plot

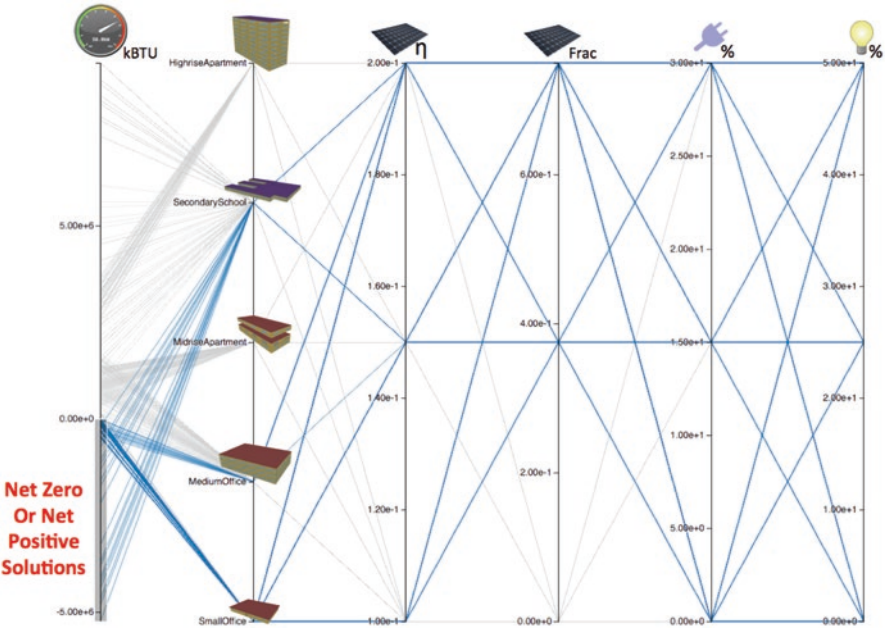

Fig. 7.37 Net zero building study with output filtering

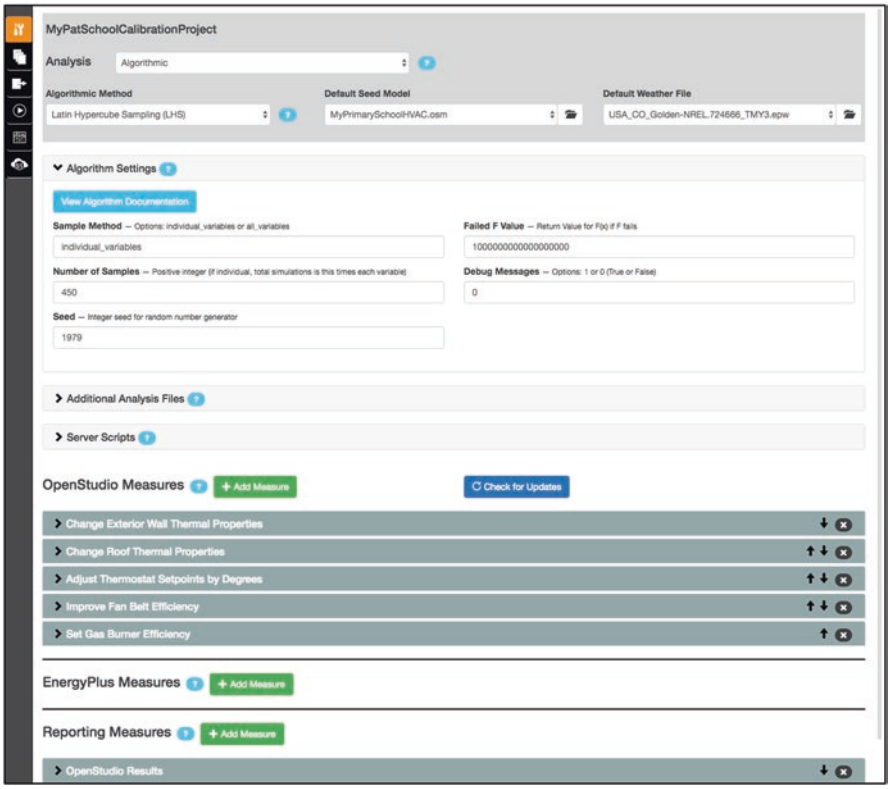

Fig. 7.38 Setting up an LHS sampling problem with our school model

calibration. While we will focus on a narrow application of these algorithms, rest assured they may be used to solve a wide range of design problems.

We will continue to work with our Model from Chap. 4. Launch PAT and begin by:

1. Creating a new project called "MyPatSchoolCalibrationProject."
2. On the Analysis ( ) Tab, set the Analysis mode to Algorithmic.
3. Set the Algorithmic Method to "Latin Hypercube Sampling (LHS)."
4. Add *MyPrimarySchoolHVAC.osm* as your default SEED Model.
5. Add USA_CO_Golden-NREL_724666_TMY3.epw as your default weather file.
6. Under algorithm settings, set the number of Samples to 75.[17]
7. Add the OpenStudio and Reporting Measures shown in Fig. 7.38.
8. Specify the Measure arguments as variables as shown in Table 7.1.
9. Proceed to the Outputs ( ) Tab and add Outputs as shown in Fig. 7.39.
10. Save your project before proceeding to the Run ( ) Tab.

[17]You may choose to use fewer samples for faster simulations or if you have fewer workers available. Our analysis will produce 6 × 75 or 450 data points.

Table 7.1 Variable assignments for LHS school model analysis

Measure	Argument	Variable	Distribution	Default	Mean	Std dev	Min	Max
Change exterior wall thermal properties	r_value_mult	Continuous	Normal	0.9	0.9	0.1	0.5	1.5
Change roof thermal properties	r_value_mult	Continuous	Normal	0.9	0.9	0.05	0.5	1.5
Adjust thermostat setpoints by degrees	cooling_adjustment	Continuous	Triangle	0	0	1	−3	3
	heating_adjustment	Continuous	Triangle	0	0	1	−3	3
Improve fan belt efficiency	fan_eff	Continuous	Uniform	0	–	5	−40	0
Set gas burner efficiency	eff	Continuous	Uniform	0.8	–	0.1	0.6	0.9

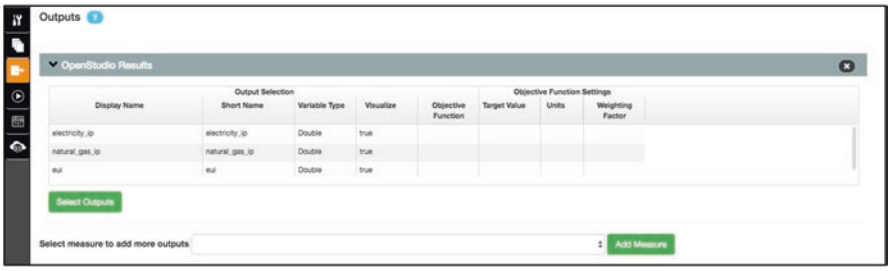

Fig. 7.39 Setting up outputs for LHS sampling problem with our school model

On the Run ( ) Tab select "Run on Cloud" and specify the Remote Server Type you will be using. Most users will select Amazon Cloud and set up their AWS run as discussed in Sect. 7.5.2. Configure your settings and connect to the server. Once connected, press the Run Entire Workflow Button to start the LHS analysis. The screenshot in Fig. 7.40 shows an analysis in process on a dedicated remote server.

> **Tip**: Before Running a large analysis, the authors always recommend testing a small version of it. You can do this by running a small number of LHS samples (1 or 2), or temporarily switching to the "Pre Flight" or "Single Run" algorithms.

Once the analysis is running, click the View Server Button to open the OpenStudio Server console in a web browser. The server's web page should look like Fig. 7.41 while the LHS analysis is in progress.

Clicking the View Analysis Button opens this particular LHS analysis' page shown in Fig. 7.42. This particular screenshot shows the page after all 450 data points have

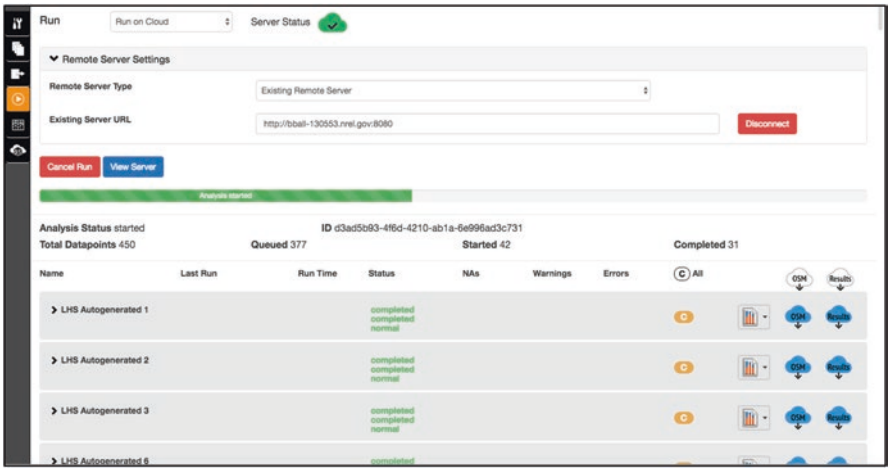

Fig. 7.40 LHS sampling analysis in progress for school model problem

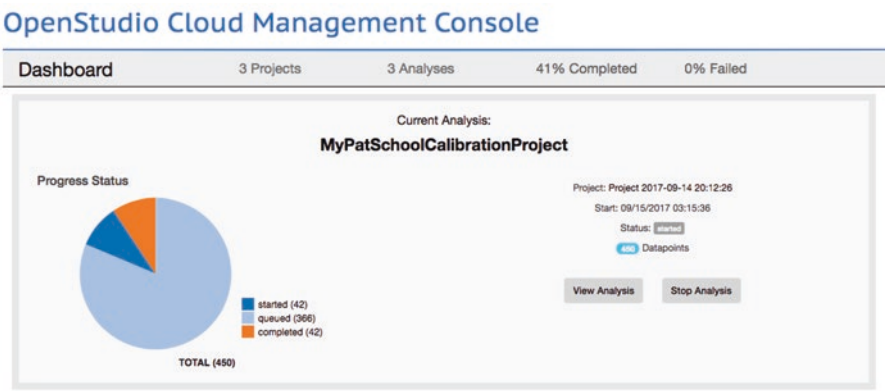

Fig. 7.41 OpenStudio Server reporting progress of LHS analysis for school model problem

been simulated, however interim results and plots may be available while the analysis is in progress. One such result is the variable summary shown in Fig. 7.43. This page includes plots allowing us to verify the variable distributions we requested.

Once the LHS analysis is complete, take some time to explore the visualizations and data point information available via the links shown in Fig. 7.42. In addition to the individual data points, which include the Model files, EnergyPlus results, HTML reports, and more; the CSV and R Data Frame Results are useful downloads from the main analysis page that allow you to further explore the relationships between these inputs and outputs.

One particularly useful feature of OpenStudio Server is illustrated in Fig. 7.44, the parallel coordinate plot. As discussed in Sect. 7.6, this interactive tool allows the user to apply filter ranges to both the inputs and outputs to quickly identify a

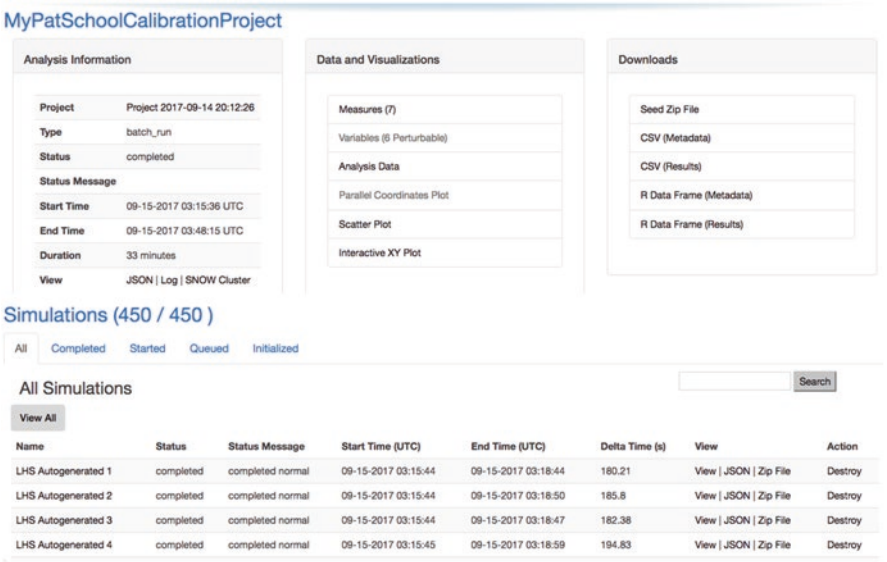

Fig. 7.42 LHS analysis report page for school model problem

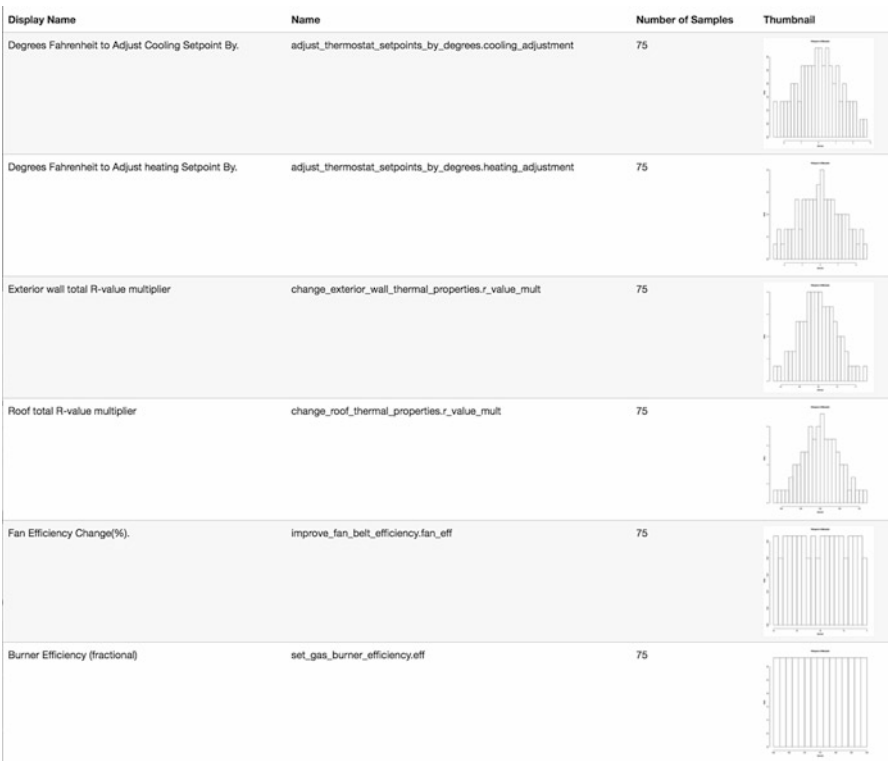

Fig. 7.43 Variable distributions for LHS sampling of school model problem

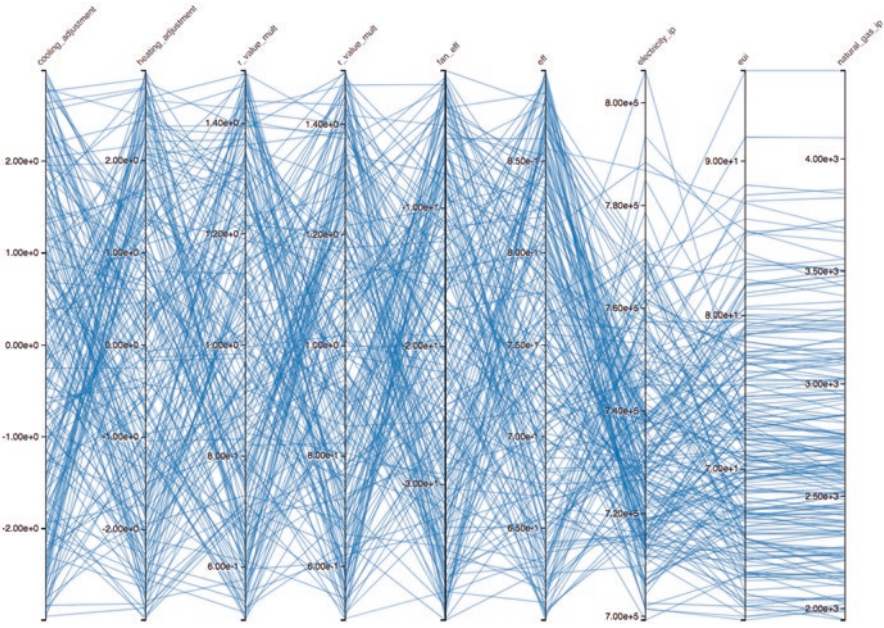

Fig. 7.44 Parallel coordinate plot for LHS sampling of school model problem

particular data point out of the 450 we simulated that may be of interest. Without tools like this, finding solutions of interest can be a bit like looking for a needle in a haystack.

The LHS analysis we just completed contained a number of Measures related to the school's insulation, temperature setpoints, fan and gas burner efficiencies. The variable ranges we selected reflect a range of uncertain values that affect the energy performance of the building. For example, R value multipliers less than one reduce the effectiveness of the school's insulation, driving up energy consumption. Lower fan and burner efficiencies also degrade the performance of the building. The study produced a population of 450 schools reflecting variation in six dimensions. Obviously, this can be extended to encompass more characteristics related to school construction, Space loads, occupancy schedules, and HVAC systems. As more degrees of freedom (Measure variables) are added to such an analysis, the variation in energy performance will continue to increase.

Previous chapters have treated our primary school as a new building with Measures used to model design features that improve performance. However, we can also use Measures to approximate degraded performance due to host of circumstances like aging, poor construction, commissioning, or operation. This leads us to the next exercise where we will consider the hypothetical case that our school is not a new building but intended to model an existing school. The fundamental question we will attempt to answer: "If our school is best described by one of the 450 we simulated above, which one is it most likely to be?"

7.8 Checkpoint Eleven: Calibration via Sensitivity Analysis and Optimization

Modeling existing buildings presents different challenges than analyzing new building designs. When modeling for new construction, the design team and analyst may have significant leeway in changing the form, fabric, and sometimes even the function of a building, its Spaces, and systems. Aside from general performance goals, there are likely few expectations for the exact energy performance of a building, and comparative assessment of EE Measures is sufficient to drive design decisions; not so with an existing building. When faced with a decision to install a Measure, the owner of an existing building owner has existing utility bills as a benchmark of performance, and a natural expectation that EE investment will reduce those bills sufficiently to pay for themselves over some period of time. This raises the bar for the accuracy of modeled predictions and brings us to the topic of Model calibration.

Model calibration is the process of adjusting a Model's inputs to align the outputs with measured data. While we could consider measuring many aspects of building performance and "tuning" a Model to best represent them, we are primarily concerned with energy consumption. Energy consumption data may be provided in the form of monthly utility billing data, interval data from so-called "smart meters," or even finely grained time-series information recorded via data logger. OpenStudio has a means of adding interval consumption data to a Model for the purpose of calibration.

Pretend that our school was actually built in 1992 in Golden, CO. We have monthly electricity and gas consumption billing data from our local utility for the year 2017. Further, we have an EPW that represents the Actual Meteorological Year (AMY) weather for that year.[18] Open your trusty MyPrimarySchoolHVAC Model in the OpenStudio Application once more. On the Site ( ) Tab make sure you are using the same weather file you used in the previous exercise. In the section labeled "Select Year by," check the Calendar Year box and set the year to 2017.

Next click on the Utility Bills Sub-Tab and use the Button to add an Electric Utility Bill to your Model. Name the Bill "Electric," and add new billing periods as needed. Enter the monthly consumption data as shown in Figs. 7.45 and 7.46.

Add a second bill named "Gas" with the monthly consumption data shown in Fig. 7.47. Take care to enter the gas consumption data using the correct units of MBtu. Save your Model when you are finished and exit the OpenStudio Application.

So now we've added the "measured" data that we will judge how well calibrated our Model is. The next question we must ask ourselves, what variables do we need to modify to perform the calibration? As we have seen, a complete energy Model contains many thousand input parameters. Where do we even start knowing which ones to modify to align modeled and measured performance?

[18] It is important to note that model calibration should always be performed with AMY data corresponding to the period that the available calibration data was obtained. Using TMY data to calibrate a model is like mixing apples and avocados. That said, our "utility billing data" was generated from TMY weather conditions, so it is appropriate to use that weather file in this exercise.

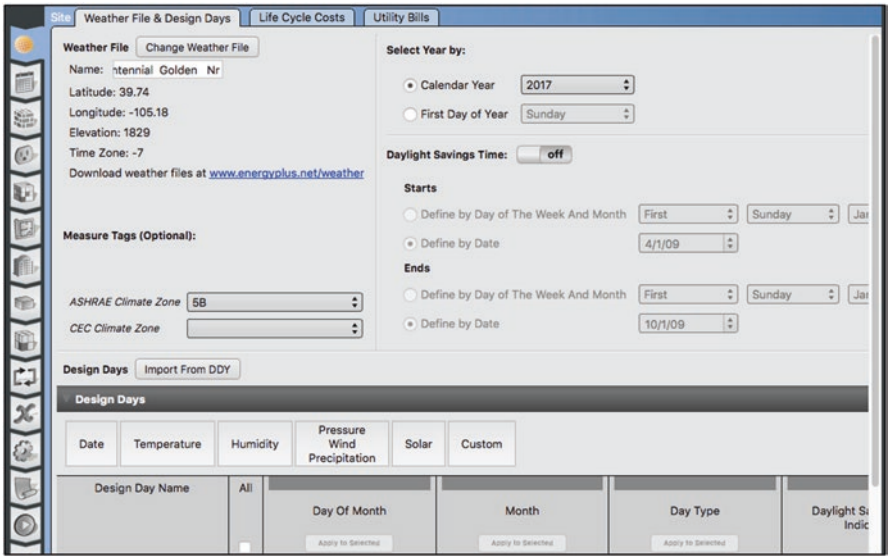

Fig. 7.45 Preparing to add electricity and gas consumption data to school model

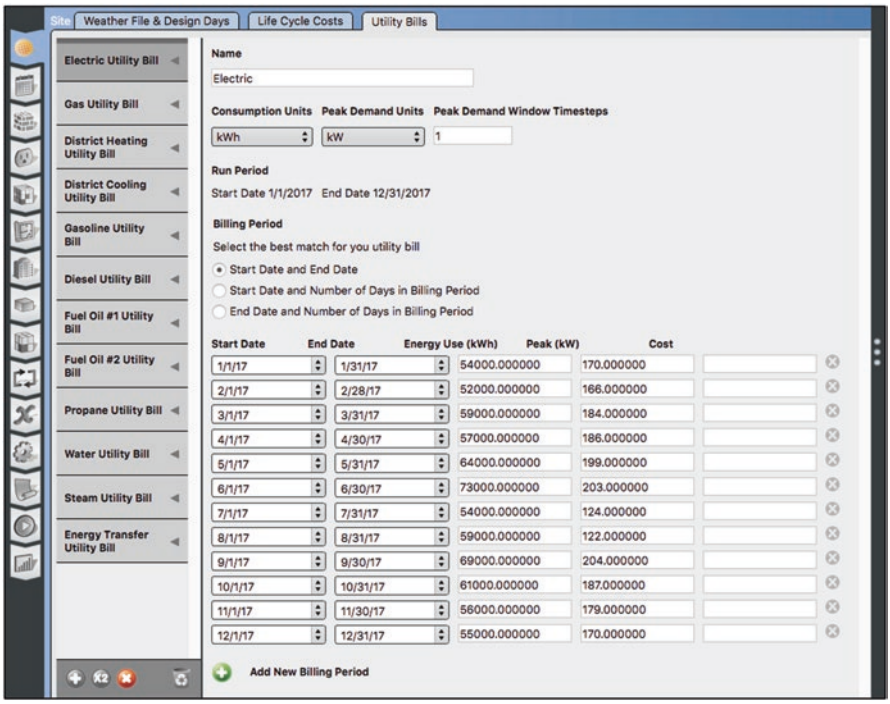

Fig. 7.46 Adding electricity consumption data to school model

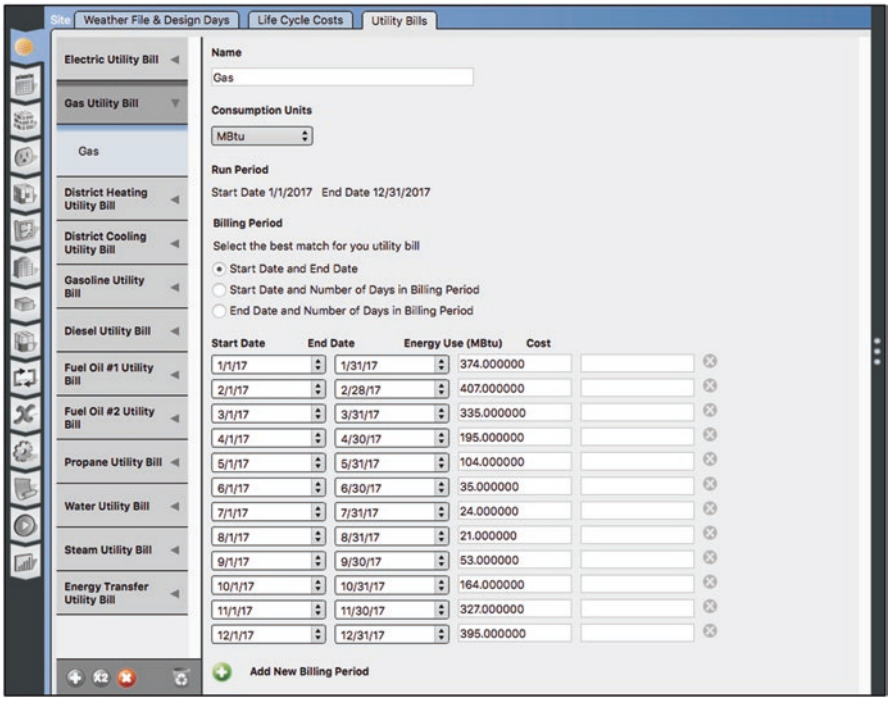

Fig. 7.47 Adding gas consumption data to school model

When modeling a real building, there are things that are easy to know. What are the dimensions of the envelope and Spaces? What are the hours of operation? How many computers are installed in classroom, and how much power do they consume? What type of mechanical systems are installed for heating and cooling? Whenever possible, the modeler should use known data to inform inputs. Calibration variables are best suited to parameters the modeler is uncertain about. Equipment efficiencies, effective R values, infiltration rates, etc. are rarely known quantities. Nevertheless, one could claim that there are still hundreds of uncertain parameters in any given building. The next algorithm we will examine, the Morris Method, is very helpful in determining which variables most affect aspects of a building's performance. This allows us to identify and remove those variables that have little impact, while retaining those that matter.

In PAT, replace the old Seed Model with the one we just modified. You can manually copy the OSM into your PAT project's seed sub-folder or you can use the file menu in PAT to select the revised Model. On the Analysis ( ) Tab switch the Algorithm from LHS to Morris Method and set the Algorithm Settings as shown in Fig. 7.48.

Launch PAT and:

1. Load the "MyPatSchoolCalibrationProject."
2. Set the Algorithmic Method to "Morris Method."

3. Add your **newly modified** *MyPrimarySchoolHVAC.osm* as your default SEED Model.[19]
4. Under algorithm settings, set r to 10 and Levels to 30 as shown in Fig. 7.48.
5. Add the "Calibration Reports Enhanced" Reporting Measures shown in Fig. 7.48.
6. Proceed to the Outputs ( ) Tab and add Outputs as shown in Fig. 7.49.[20]
7. Save your project before proceeding to the Run ( ) Tab.

The most significant changes in the Project include the choice of algorithm and the addition of "performance objectives" that are computed by the Calibration Reports Measure using the consumption data we added to the Model. Among other things, the Measure produces two primary quantities that are indicative of the "goodness of fit" for a given simulation run. They are the net mean bias error (NMBE) and coefficient of variation of the root mean squared error (CVRMSE). These are defined according to:

$$NMBE = \frac{100}{\overline{y}} \times \frac{\Sigma\left(y_i - \hat{y}_i\right)}{n} \quad CVRMSE = \frac{100}{\overline{y}} \times \sqrt{\frac{\Sigma\left(y_i - \hat{y}_i\right)^2}{n}}$$

where:

y_i is the ith measured data point,
$\hat{y}_i$ is the ith simulated data point,
$\overline{y}$ is the mean of the measured data points, and
n is the total number of data points.

For our analysis, the Measure will produce NMBE and CVRMSE values based on the errors in monthly electricity and gas consumption resulting in four different performance objectives. Ideally, these will be zero for perfectly matched analytical and empirical values. In practice, these values can be minimized, but will never equal zero due to measurement and modeling errors. ASHRAE has published Guideline 14, which prescribes tolerances for NMBE and CVRMSE for building calibration.[21] A "good" calibration will have a NMBE less than 10% and CVRMSE less than 30%. The Calibration Reporting Measure also checks against these threshold values and reports pass/fail results. We will make use of these metrics for both the Morris Method and subsequent optimization of variables that minimize them.

Connect to the OpenStudio Server and run the Morris Method analysis. As before, you can open the Server window in a web browser to monitor the process and interact with interim results. The completed analysis should look like Fig. 7.50. The Morris Method is an algorithm that offers an additional download on the Project's page called *Algorithm Results.zip*. Download this file and examine its contents.

[19] You can manually copy the OSM into your PAT project's seed sub-folder or you can use the file menu in PAT to select the revised model, but **do not forget this step**.

[20] You may choose to leave the OpenStudio Results outputs in your project. We have removed them to minimize server reporting clutter.

[21] ASHRAE Guideline 14–2014: Measurement of Energy, Demand and Water Savings, ASHRAE, 2014.

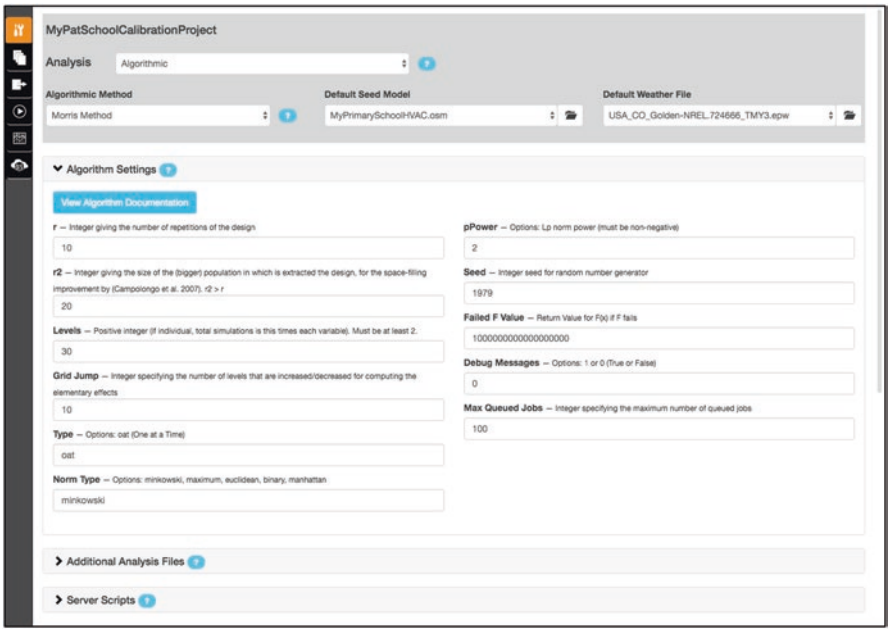

Fig. 7.48 Setting up a Morris Method problem with our school model

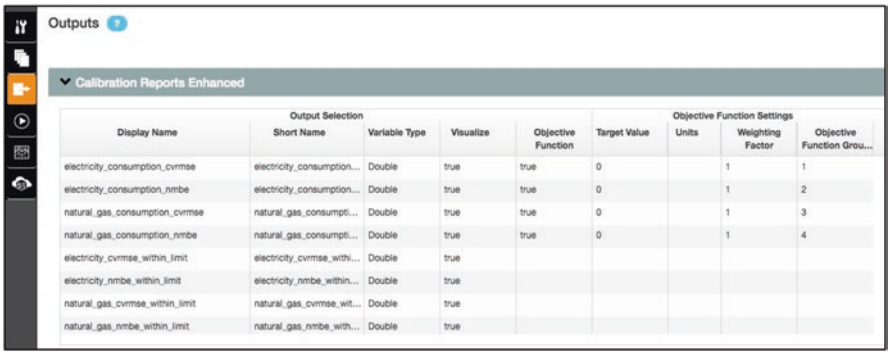

Fig. 7.49 Setting up outputs for Morris Method problem with our school model

Algorithm Results.zip contains a number of pre-generated plots based on your choice of variables and the NMBE and CVRMSE objective functions. Figures 7.51 and 7.52 contains two sets of four plots related to gas and electric NMBE and CVRMSE. Figure 7.51 shows the "elementary effects" of each input variable on the output of interest. μ^* shown in Fig. 7.52 is the mean value of the absolute value of the elementary effects and represents the relative sensitivity of an output of interest to all input variables. Variables with comparatively large μ^* values are very significant in shaping an output, whereas μ^* equal to zero has no relationship.

Fig. 7.50 Completed Morris Method analysis for school model problem

A few observations for our school Model are apparent. First, the most significant variables impacting gas consumption are not necessarily the same as those effecting electricity use. For example, the cooling setpoint change was the biggest driver of electricity consumption variability. For gas consumption, the heating setpoint was the most significant variable. Does this make sense?

Second, the burner efficiency has no impact on either gas or electricity performance metrics. This might seem surprising but open up the seed Model and examine its HVAC system to see if this outcome is reasonable. Assuming you have satisfied yourself that it has no impact, this variable (and Measure) may be removed from our Project.

What about the other variables? If we were only concerned about gas consumption, we might consider removing cooling setpoint as a source of uncertainty in our calibration. However, this would have a serious impact on our ability to shape the electricity consumption of our Model. The converse is true for the heating setpoint, so we will leave both of these in our project. Again, looking at gas consumption, one

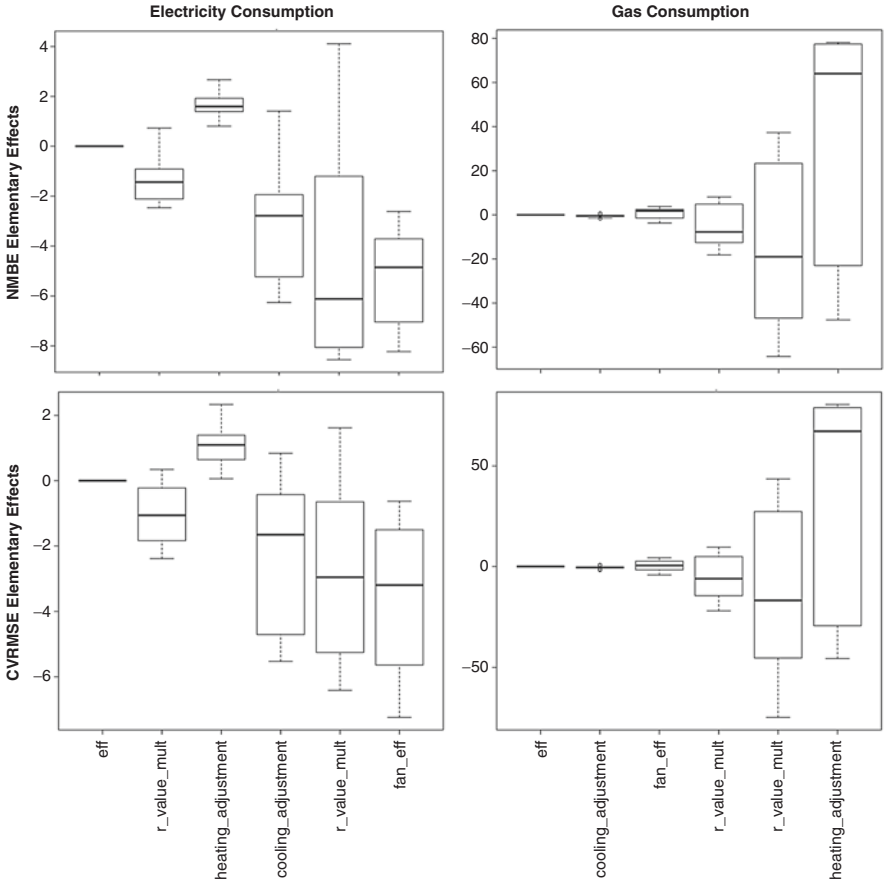

Fig. 7.51 Morris Method elementary effects box plots for school model problem

might consider removing fan efficiency as a variable. Once again though, this would have a detrimental effect on our ability to shape electric consumption, as it is the second most significant elementary effect. Based on these considerations, we are probably wise to only prune the burner efficiency Measure from our calibration.

We are now going to shift from performing a sensitivity analysis of our four performance metrics to allowing SPEA2 to utilize the remaining Five variables to minimize NMBE and CVRMSE for gas and electricity consumption. Back in PAT:

1. Delete the burner efficiency Measure from the Project.
2. Set the Algorithmic Method to "SPEA2."
3. Under algorithm settings, set Number of Samples to 40[22] and Generations to 2 as shown in Fig. 7.53.
4. Save your project before proceeding to the Run ( ) Tab.

[22] Algorithms like SPEA2, PSO, Rgenoud, etc. will make the most cost-effective use of your server and workers by setting the number of samples to be equal to the number of worker nodes available. This minimizes the number of workers that sit idle between optimizer iterations.

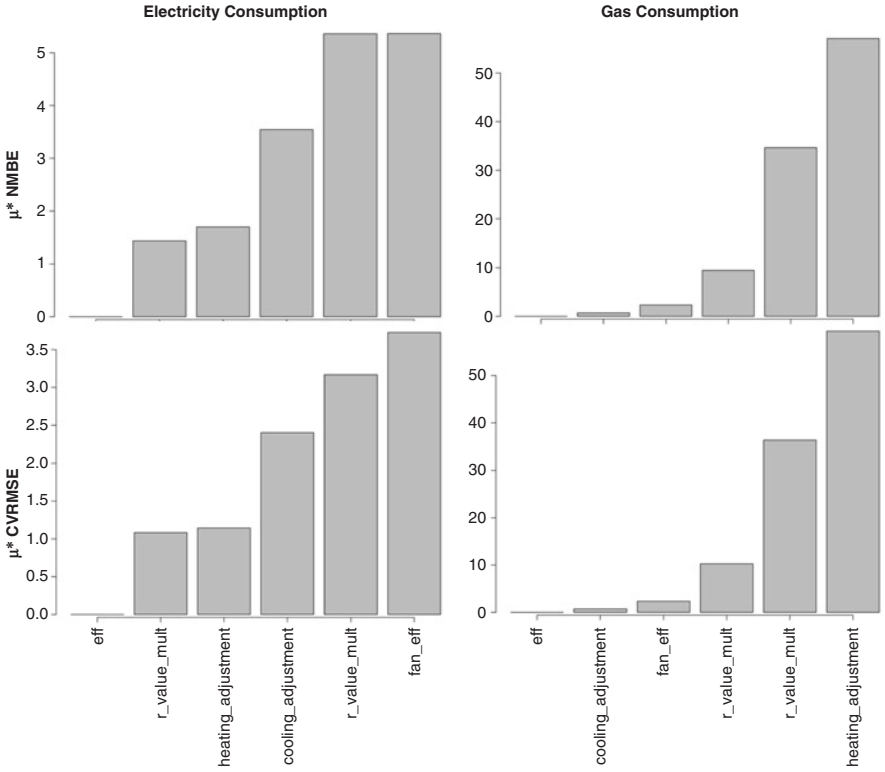

Fig. 7.52 Morris Method μ* plots for school model problem

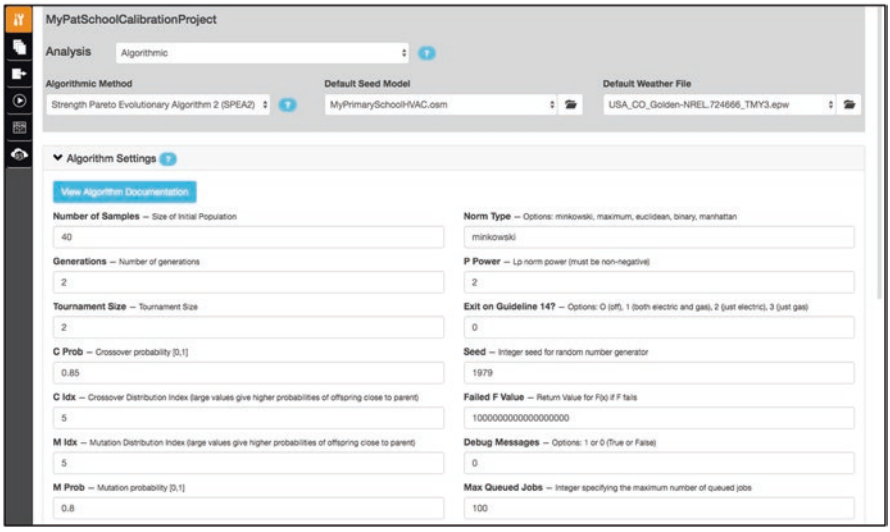

Fig. 7.53 Setting up an SPEA2 optimization problem with our school model

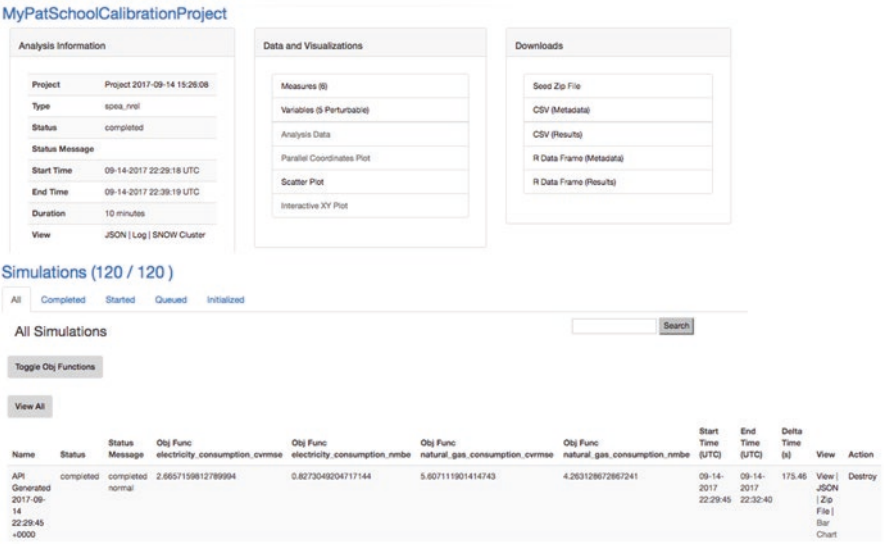

Fig. 7.54 School model calibration project with completed SPEA2 optimization

Run the analysis and jump over to the OpenStudio Server console in your web browser (Fig. 7.54). This project is smaller and faster than the previous two by virtue of having removed a variable and the fact that the algorithm is focused on achieving a specific objective rather than studying the school Model's performance space more generally. Note in this screenshot we have clicked on the Toggle Obj Functions Button just above the data point list to display the four objective functions. Scrolling through the points reveals a range of good (and bad) solutions that the optimizer discovered on its way to the "best" combination of input variables.

Figure 7.55 uses the server's interactive XY plot feature to examine a trajectory of solutions that SPEA2 explored on its way to finding minimal values of NMBE and CVRMSE for gas and electricity.[23] Out of curiosity, you might want to try some of the other optimizers to see how they perform. Figure 7.56 shows how PSO arrives at a similar solution.

We have annotated these plots with regions showing data points that satisfy ASHRAE Guideline 14. This underscores two considerations. First, the solution space that we pulled our "real school" from in Checkpoint Nine was actually fairly small, and none of the variables we used to create this experiment were wildly divergent. Therefore, it's not that hard to find an answer that satisfies Guideline 14 for this simple problem.

More importantly though is the fact that a multiplicity of solutions exist that satisfy the criteria. This is very typical of building Model calibration problems and

[23] Because of the dimensionality of the space the optimizer is traversing, we can only visualize slices of the solution in plots like these.

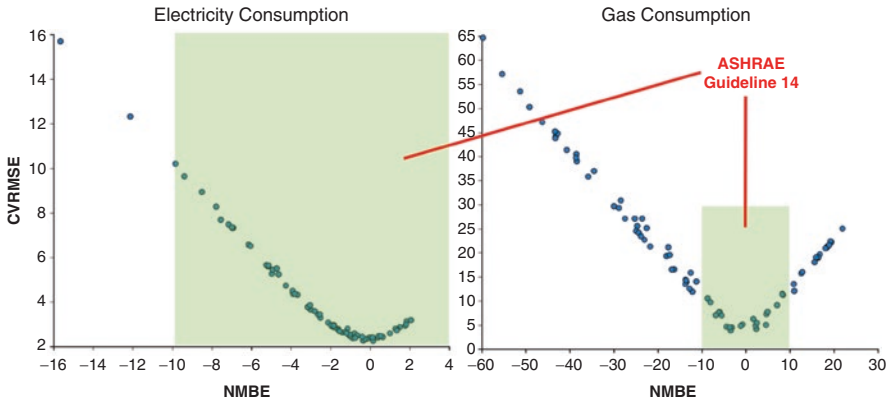

Fig. 7.55 SPEA2 optimizer results for school model calibration problem

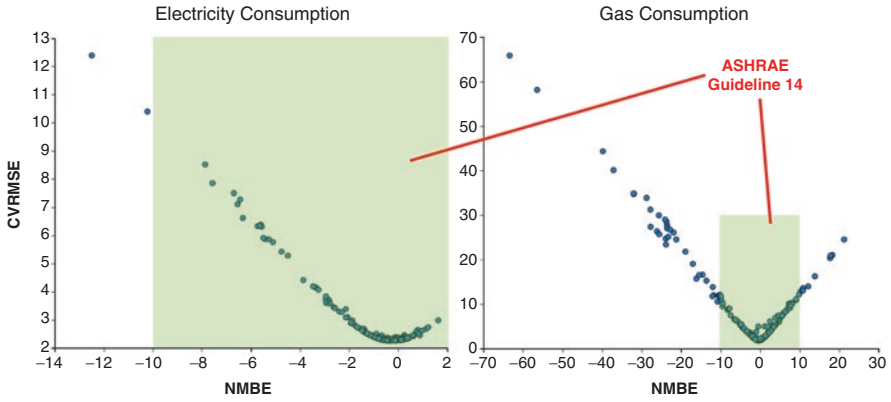

Fig. 7.56 PSO optimizer results for school model calibration problem

becomes more pronounced when the number of uncertain variables increases while the number of dependent outputs (NMBE and CVRMSE) remains very small. These are known as mathematically underdetermined problems, and they result in multiple possible solutions yielding similar outcomes. While an unimpeachable, mathematical solution may elude us; we can bring some human expertise into play to narrow the range of potential solutions.

OpenStudio Server's interactive parallel coordinate plot can be helpful in exploring solution spaces and identifying data points of interest (Fig. 7.57). You may have guessed that the four "within_limit" outputs shown on the far right of this plot correspond to solutions that achieved ASHRAE Guideline 14 electricity or gas compliance for NMBE or CVRMSE. Use your mouse to filter where these outputs are equal to one and the corresponding metric falls within tolerances (Fig. 7.58). The remaining blue "threads" represent simulation data points that fall within both green boxes shown in Fig. 7.55.

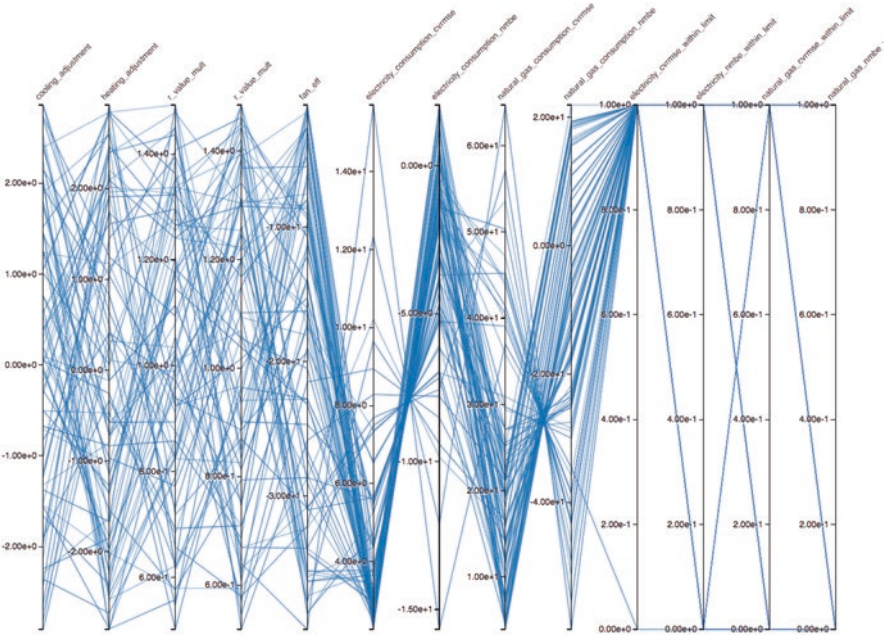

Fig. 7.57 Parallel coordinate plot of all SPEA2 data points

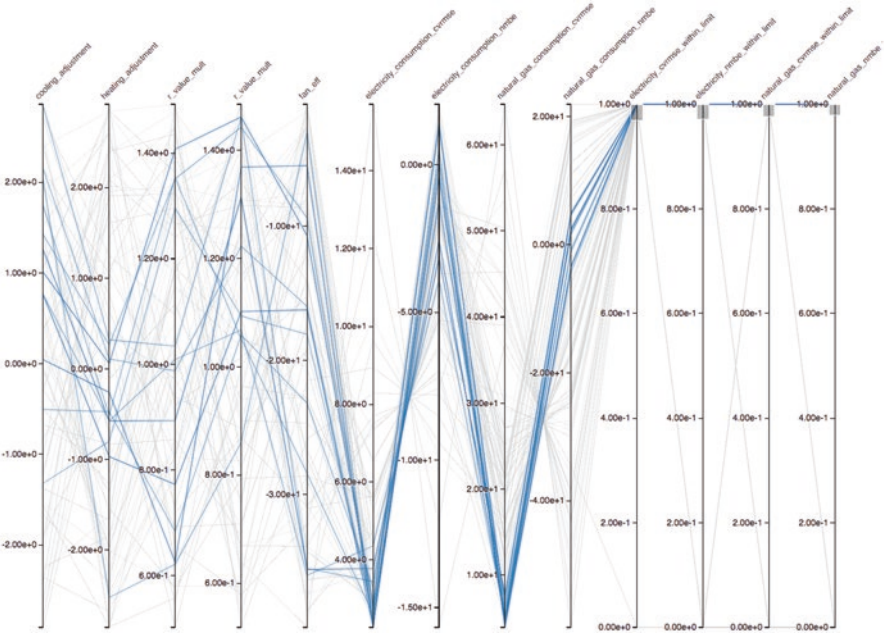

Fig. 7.58 Parallel coordinate plot of SPEA2 results filtered for ASHRAE Guideline 14

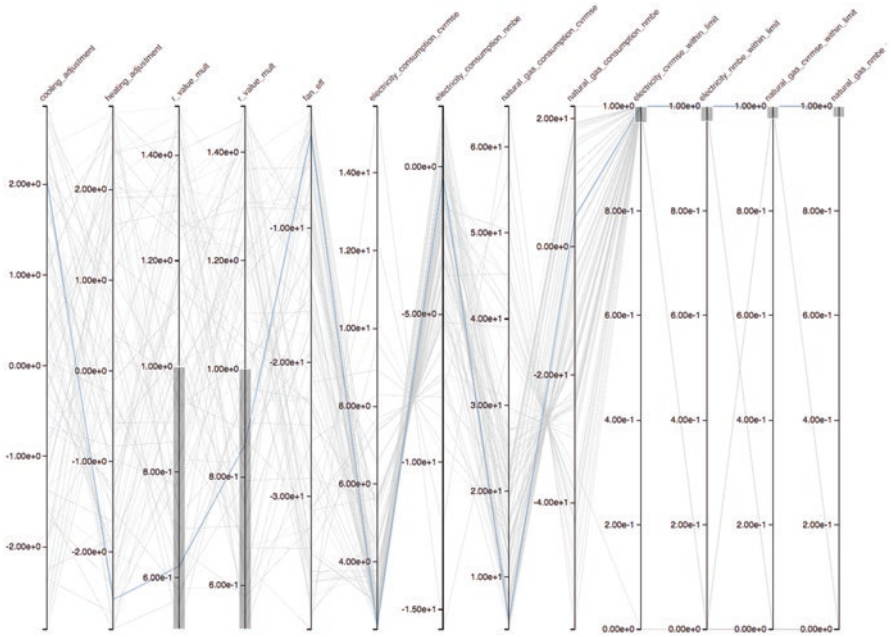

Fig. 7.59 Additional filtering of SPEA2 results on parallel coordinate plot

That helped a bit, but there are still a large number of potential solutions. To refine our search further, we need to make use of one seemingly insignificant detail that you may have overlooked at the beginning of this exercise – the school was built in 1992.

How can the vintage of the building help us narrow our search? A seasoned energy modeler would recall one of the very first assumptions that we made back in Chap. 2 when we first built our school Model. In that exercise, we assigned a Construction Set corresponding to ASHRAE 90.1-2010. This represents a more stringent level of the energy code than when our hypothetical school was built nearly two decades earlier. It's quite likely that our school has less insulation than how we modeled it. In our study, that would correspond to data points with R value multipliers less than one. Applying an appropriate filter to the wall and roof "r_value_mult" variables produces Fig. 7.59. Abracadabra!

Below the parallel coordinate plot is a list of all data points that meet the filter requirements. As we continue to filter results, the list has winnowed down to less than a handful of points. In fact, these points are identical solutions that the optimizer arrived at from different directions. Selecting one of them allows us to see the specific Measure variable values that define the data point. The data point page also allows convenient download of the specific Model, simulation results, and associated reports shown in Fig. 7.60.

Output from the Calibration Report for this data point is one of the available report links and allows us to quickly visualize the performance of this particular Model relative to our school's "measured" consumption data. Figure 7.61 compares one such report for a bad solution with the SPEA2 solution that met our criteria.

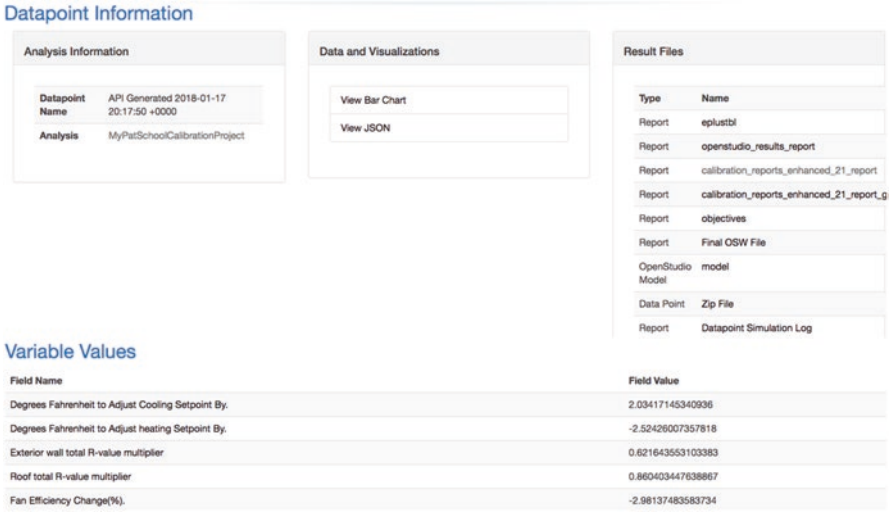

Fig. 7.60 An SPEA2 optimized solution for the school calibration problem

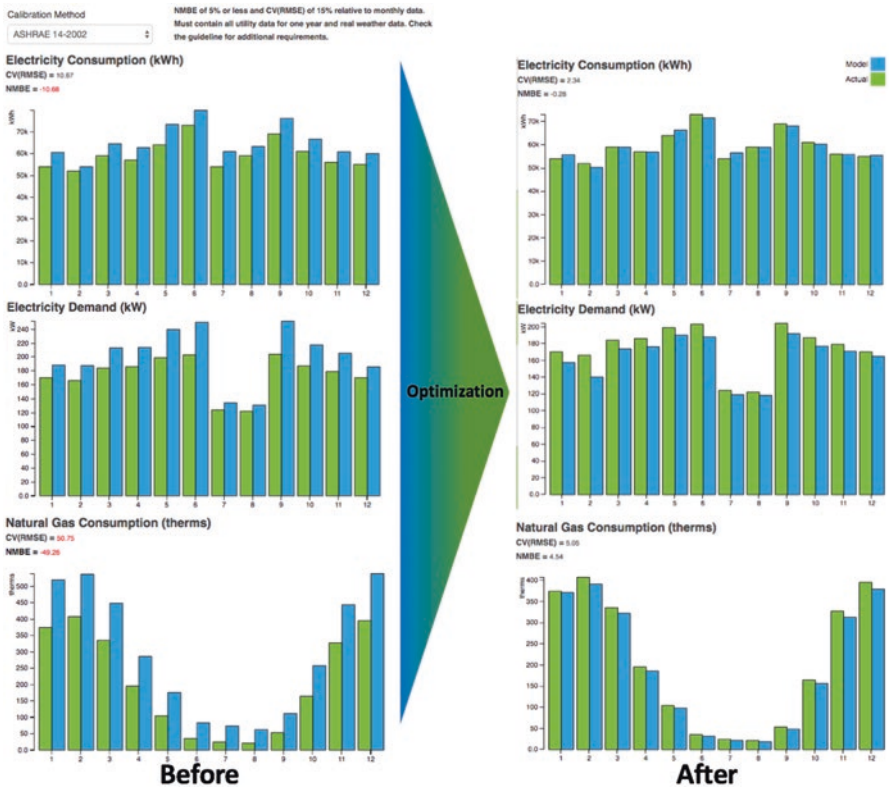

Fig. 7.61 School model calibration reports pre and post SPEA2 optimization

This exercise attempted to provide some experience and insight in using PAT and OpenStudio Server to solve a common problem that faces building modelers focusing on retrofit of existing buildings. Given the huge number of uncertain variables associated with any building, these are fundamentally difficult problems to solve. It is important to approach the tools with realistic expectations, and to leverage common sense, experience, and even intuition when trying to arrive at the "best" Model for a particular building.

7.9 Additional Exercises

There are a number of potential exercises involving the Model you created for "Additional Exercises" in Chap. 4. A few to consider include:

1) Learn more about uncertainty in predicted Model performance.
   - Consider sources of uncertainty in your model and select Measures for the BCL that allow you to modify related inputs: e.g. electric load power, infiltration, HVAC system performance, etc.
   - Create continuous or discrete variables for key parameters in these Measures and use them in an LHS problem to see how variation impacts annual energy use and peak power consumption.
   - Download R or Excel data from OpenStudio Server to plot EUI distributions for the Models in your sample population.
2) Identify some of the key parameters your Model is most sensitive to.
   - Expand upon the previous analysis by using Morris, Sobol, or FAST99 to identify parameters that significantly impact annual energy use, peak power consumption, or other outputs of potential interest.
3) Calibrate your Model using available consumption data.
   - Obtain electric or gas consumption data from the building owner or facility manager.
   - Obtain an AMY weather file for your region.
   - Calibrate your Model using the process described in Checkpoint Eleven.
4) Optimize efficiency Measures for your calibrated Model.
   - Set up an objective function[24] to assess the relative goodness of a collection of efficiency measures.
   - Select Measures and input ranges for your optimization problem.
   - Experiment with one or more optimization algorithms and compare the solutions they produce.

[24] Give some thought to definition of an objective function. For example, optimizing on annual energy use alone will result in a building with no windows, maximum insulation, ultra-high efficiency HVAC systems, etc. Add additional objectives (e.g. cost, comfort, or competing factors like heating and cooling energy) to create more realistic optimization problems.

References

Barricelli N (1957) Symbiogenetic evolution processes realized by artificial methods. Methodos 9:143–182

Belding T (1995) The distributed genetic algorithm revisited. In: Eshelman LJ (ed) Proceedings of the sixth international conference on genetic algorithms. Morgan Kaufmann, San Francisco, CA, pp 114–121

Deb K, Pratap A, Agarwal S (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197

Ephramac (2017) 2D-Partikelschwarm sucht globales Minimum, 12 Jan 2017. https://commons.wikimedia.org/wiki/File:ParticleSwarmArrowsAnimation.gif

Fraser A (1957) Simulation of genetic systems by automatic digital computers. Aust J Biol Sci 10:484–491

https://www.r-project.org/

Kennedy J, Eberhart R (1995) Particle swarm optimization. Proc IEEE Int Conf Neural Netw IV:1942–1948

King R, Deb K, Rughooputh H (2010) Comparison of NSGA-II and SPEA2 on the multi-objective environmental/economic dispatch problem. Univ Mauritius Res J 16:485–511

McKay M, Beckman R, Conover W (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21:239–245

Mebane W Jr, Sekhon J (2011) Genetic optimization using derivatives: the RGenOUD package for R. J Stat Softw 42(11):1–26. https://www.jstatsoft.org/article/view/v042i11

Morris M (1991) Factorial sampling plans for preliminary computational experiments. Technometrics 33:161–174

Nelder J, Mead R (1965) A simplex algorithm for function minimization. Comput J 7:308–313

Saltelli A, Tarantola S, Chan K (1999) A quantitative, model independent method for global sensitivity analysis of model output. Technometrics 41:39–56

Sobol I (2001) Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. Math Comput Simul 55:271–280

Zitzler E, Thiele L (1998) An evolutionary algorithm for multi-objective optimization: the strength Pareto approach. Technical report 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich