Design Optimization with GenOpt®

Michael Wetter
Simulation Research Group
MWetter@lbl.gov

Abstract

Although simulation is increasingly being used in building system design, the full potential of simulation is usually not achieved. To improve system performance, designers generally guess different values of system parameters and redo the simulation. This is inefficient and labor intensive. Also, if the number of parameters being varied exceeds two or three, the designer can be overwhelmed in trying to understand the nonlinear interactions of the parameters. However, techniques exist that allow automatic, multidimensional optimization of a simulation model, leading to better design with less effort.

We describe how such optimization can be done using GenOpt, a generic optimization program. We show an example of how to use GenOpt to design an office building such that source energy consumption for heating, cooling, and lighting is minimal with respect to selected design parameters.
In this example, the optimization yields 14% energy savings. The additional time required to set up the optimization is about an hour. The measures found by using optimization not only decrease operating costs, but also lead to better daylighting usage, which results in higher comfort for the building occupants.

1 Introduction

Setting up an input file for a thermal building simulation is time intensive. Once the analyst has set up the input files, she/he does not utilize the potential that computer simulation offers in optimizing the design. Today, "optimizing" a building or HVAC system consists of guessing parameter settings that may lead to better system performance. This is very time consuming. Also, if the number of parameters being varied exceeds two or three, the analyst has difficulty understanding the interaction of the system parameters in order to make an educated guess that leads to further improvement. Hence, only limited improvement is achieved.

However, mathematical optimization allows determining the values of system parameters that lead to optimal system performance using a computational procedure. LBNL's Simulation Research Group developed GenOpt, a generic optimization program, that allows such optimization. Doing the optimization requires only little additional labor compared to preparing the simulation input. GenOpt can be used with any simulation program that has text based I/O, such as EnergyPlus, DOE-2, SPARK[1], BLAST [2], TRNSYS[3], etc. or any user-written code.

[1]For more information on EnergyPlus, DOE-2, and SPARK, see `http://SimulationResearch.lbl.gov`
[2]For more information on BLAST, see `http://www.bso.uiuc.edu`
[3]For more information on TRNSYS, see `http://sel.me.wisc.edu/TRNSYS`

Our goal here is to show step by step how an optimization with GenOpt 1.1 can be done on a typical office building, and how optimization improves building design. The simulations will be done with EnergyPlus 1.0β4, and we will minimize annual source energy consumption for heating, cooling, and lighting.

We will first introduce some terminology, and explain the building being optimized, before we outline the procedure that allows us to perform the optimization. In the last section, we will present the results of the optimization.

2 Terminology

By solving a minimization problem, a set of **independent parameters** (also called **design parameters** or **free parameters**) will be modified to find a minimum of a function. We will denote this set by $x \in \mathbb{R}^n$. The independent parameters are typically values such as insulation thickness, window sizes, volume flow rate, etc. Clearly, they cannot take on any value, since, e.g., a window area cannot be negative. Therefore, the independent parameters are constrained to a **feasible set**. In building simulation, most parameters can be constrained by specifying a lower and upper bound. These kinds of constraints are called **box constraints**. The function being minimized is called the **objective function**. It typically stands for a value such as annual energy consumption, operating cost, peak load, etc.

The set of independent parameters that minimizes the objective function is called **minimizer** (or minimum point). A minimizer may be a **local minimizer** or a **global minimizer**. It is generally impossible to know whether a local minimizer is also a global minimizer. However, finding a local minimizer is already an improvement in system design compared to the initial values of x.

To solve an optimization problem, an optimization algorithm is initialized with some parameter set that is in the feasible set. The algorithm then selects iteratively new parameter sets such that a minimizer is found without having to evaluate the objective function on the whole feasible set. This is usually done by checking the behavior of the function locally around a point, and by applying some strategy – based on the local function shape and previous iterations – which hopefully leads to a large step towards a minimizer.

3 Optimization Problem

We will optimize the yearly source energy consumption for heating, cooling, and lighting of an office building in Chicago. It is assumed that the energy consumption of the thermal zone shaded in Figure 1 is representative for the overall building energy consumption for heating, cooling, and lighting. In Figure 1, floor, ceiling, and north and south wall are adiabatic. The exterior walls have a U-value of $0.25\,\mathrm{W/(m^2\,K)}$ and consist of (listed from outside to inside) 1 cm wood siding, 10 cm insulation and 20 cm concrete. The ceiling and floor consist of carpet, 5 cm concrete, insulation and 18 cm concrete. Interior walls are 12 cm brick. Both windows are low-emissivity, double pane with Krypton gas fill and exterior shading device. The shading device is actuated only during summer when the solar radiation on the window exceeds $200\,\mathrm{W/m^2}$. Both windows have a fixed shading overhang of 1 m depth. The zone's daylighting control dims the light to an illumination setpoint of 500 lux at a distance of 3 m from each window. The heating set point is 20°C between 6 AM and 11 PM and 18°C otherwise. The cooling set point is a constant 25°C.

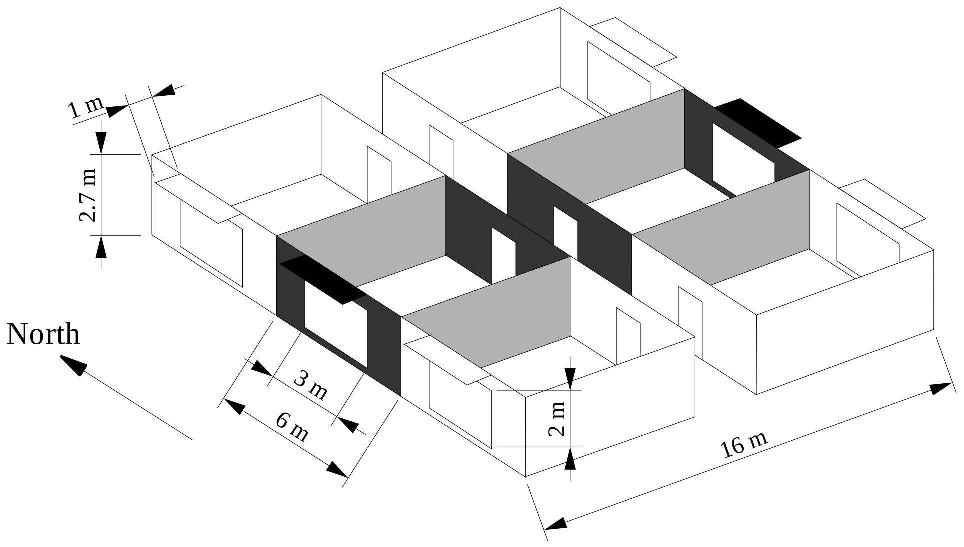


Figure 1: Optimized office zone in its initial configuration

The yearly source energy consumption is calculated by

$$E_{tot}(x) = \frac{Q_{heat}(x)}{\eta_{heat}} + \frac{Q_{cool}(x)}{\eta_{cool}} + 3\, E_{lights}(x), \tag{1}$$

where Q denotes the zone's yearly heating or cooling load, E_{lights} the zone's electricity consumption for lighting, and the efficiencies $\eta_{heat} = 0.44$ and $\eta_{cool} = 0.77$ are typical plant efficiencies that relate the zone load to the source energy consumption required for heating and cooling generation, including electricity consumption for fans and pumps. Lighting (as well as electricity for fans and pumps) is weighted by a factor 3 to convert electricity to fuel energy usage. η_{heat} and η_{cool} are obtained from [HF99] and are typical of large office buildings in Chicago.

The independent parameters are the building azimuth, α, the width of the west window, w_w, the width of the east window, w_e, and the transmittance of the movable exterior shading device, τ (assumed to be the same for solar and visible radiation). Table 1 shows the lower bounds, l, the initial values, x_0, and the upper bounds, u, of the independent parameters. The bounds on the window width have been chosen such that the window is not wider than the wall. Since the objective function is periodic in the building azimuth, we did not constrain α.

How much progress can be achieved by using optimization relative to the initial values, x_0, certainly depends on how good the initial conditions already are. For our test case, we selected x_0 such that we believed our building design is optimal with respect to $E_{tot}(x)$. The window area has been selected such that electricity consumption for lighting can be kept small due to large daylighting use. We believe that further increasing the window size would lead to higher cooling loads due to increased solar gains, without increasing daylighting consumption significantly. Also, since heating energy consumption is usually not a dominant factor for office buildings, increased solar gains in winter would not reduce the total energy consumption significantly. Further, it turned out that rotating the building by 90° while keeping the other parameter fixed decreases $E_{tot}(x)$ only by 2%. Therefore, we left the initial condition for the azimuth at 0°, as in the first guess of the design.

	l	x_0	u
building azimuth α [DEG]	$-\infty$	0	$+\infty$
width west window w_w [m]	0.1	3	5.9
width east window w_e [m]	0.1	3	5.9
shading device transmittance τ	0.2	0.5	0.8

Table 1: Lower bound, l, initial value, x_0, and upper bound, u, of the four independent parameters

As in all building simulations, equation (1) is discontinuous with respect to the independent parameter, x. Furthermore, for some values of x, the gradient of (1) with respect to x tends to infinity. Hence, we cannot use a gradient based optimization algorithm. We will therefore use a so-called pattern search algorithm developed by Hooke and Jeeves [HJ61] which does not use the gradient of (1).

4 Setting up the Optimization Problem

We assume that the simulation input files have already been written. We are now faced with the task of (i) specifying what parameters of the input files have to be varied within which range, (ii) what number in the output file has to be minimized, (iii) what optimization algorithm has to be used, (iv) how the simulation program is to be started, and (v) where the input files and output files are located.

While we outline all necessary steps to set up an optimization run, we have to refer to the GenOpt manual for a more detailed explanation of the various keywords, as well as for the mathematical description of the optimization algorithm. Based on Figure 2, we will first outline how GenOpt performs the simulation runs: GenOpt reads the **simulation input template files**, which are nothing but the simulation input files where the numerical values of each independent parameter was replaced by its variable name. GenOpt then replaces these variable names by numerical values (which were determined by the chosen optimization algorithm) and writes the **simulation input files**. Afterwards, GenOpt starts the simulation program, waits until the simulation is completed, and then reads the **simulation log files** and the **simulation output files**. The simulation log files indicate whether the simulation had an error. If not, GenOpt searches the objective function value in the simulation output files. An **initialization file** tells GenOpt where all files are located. The **command file** lists the independent parameters and their bounds. It also specifies what algorithm should be selected from GenOpt's algorithm library to perform the optimization. The **configuration file** tells GenOpt what messages in the log file indicate a simulation error, and how GenOpt has to start the optimization.

4.1 Specification of Independent Parameters

To specify what values of the simulation input files are independent parameters, we need to replace each numerical value that has to be optimized with a variable name. To distinguish variable names from other text in the input files, each variable name needs to be enclosed in percentage signs. To do so, we make a copy of the **simulation input file**, which we will call **simulation input template file**. Then, if a section of the simulation input file looks like

Input Files

initialization:	Specification of file location (input files, output files, log file, etc.)
command:	Specification of parameter names, initial values, bounds, optimization algorithm, etc.
configuration:	Configuration of simulation program (error indicators, start command, etc.)
simulation input template:	Templates of simulation input files

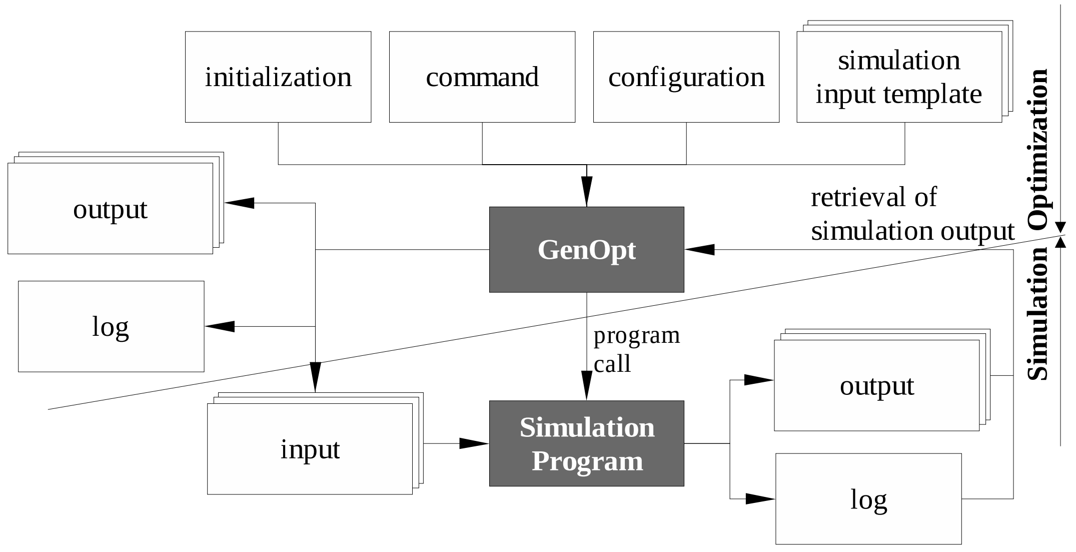


Figure 2: Interface between GenOpt and the simulation program that calculates the objective function

```
    office zone,   ! Building Name
             90,   ! Building Azimuth
              1,   ! Building Terrain
```

and we want to vary the building azimuth, we need to modify the section according to

```
    office zone,   ! Building Name
      %azimuth%,   ! Building Azimuth
              1,   ! Building Terrain
```

`azimuth` will now be the name of one of the independent parameters. The same name has to be specified in the **optimization command file**. The specification of all independent parameters listed in Table 1 looks like

```
Parameter{ Name = azimuth;  Min = SMALL; Ini = 0;   Max = BIG; Step =  10; }
Parameter{ Name = w_we_win; Min = 0.1;   Ini = 3;   Max = 5.9; Step = 0.2; }
Parameter{ Name = w_ea_win; Min = 0.1;   Ini = 3;   Max = 5.9; Step = 0.2; }
Parameter{ Name = tau;      Min = 0.2;   Ini = 0.5; Max = 0.8; Step = 0.1; }
```

The `Parameter` section shown above specifies for each variable the minimum value, the initial value, and the maximum value. It also declares a step size, which is a value that is used by the optimization algorithm (see the GenOpt manual for more information on step size).

4.2 Specification of Algorithm

GenOpt has a library with different optimization algorithms. In the **optimization command file**, we can specify what algorithm has to be used. A typical section looks like

```
Algorithm{ Main = HookeJeeves;  StepReduction = 0.5;  NumberOfStepReduction = 2; }
```

This specification causes GenOpt to use the Hooke-Jeeves optimization algorithm, and to pass two additional parameters to the algorithm. (See the GenOpt manual for an explanation of the parameters.)

In the **optimization command file**, we also need to specify a section `OptimizationSettings` where we set the maximum number of iterations and some other parameters that are commonly used by all optimization algorithms. It typically looks like

```
OptimizationSettings{ MaxIte = 200; MaxEqualResults = 50; WriteStepNumber = false; }
```

4.3 Specification of Simulation Program

We also need to specify how GenOpt can start the simulation program, and what messages in the simulation log file indicate an error of the simulation program. This information is stored in the **simulation configuration file**. GenOpt's installation already contains such files for various simulation programs (e.g., DOE-2, SPARK, EnergyPlus, TRNSYS, etc.). The user does not need to modify these files.

4.4 Function Value to be Minimized

Since most simulation programs write their output in text files that contain the function values and additional text, we need to specify how GenOpt can find the objective function value in the simulation output file. Consider an EnergyPlus output file that looks like

```
...
5,1,Cumulative Days of Simulation[] ! When Run Period Report Variables Requested
100,2,ZONE ONE,Lights-Electric Consumption[J] !RunPeriod
120,2,ZONE1AIR,Purchased Air Heating Energy[J] !RunPeriod
122,2,ZONE1AIR,Purchased Air Cooling Energy[J] !RunPeriod
End of Data Dictionary
1,CHICAGO IL UNITED STATES TMY2 94846,  41.78, -87.75,  -6.00, 190.00
5,365
100,9.8254090E+09
120,2.4119670E+09
122,1.0562006E+10
End of Data
```

Then, the function values that we want to read from the file are beyond the strings `5,`, `120,`, `122,` and `100,`. Therefore, we will specify in the **optimization configuration file** a section of the form

```
ObjectiveFunctionLocation{
      Delimiter1 = "5,"  ;   Name1 = "E_tot";
      Delimiter2 = "120,";   Name2 = "Q_heat";
      Delimiter3 = "122,";   Name3 = "Q_cool";
      Delimiter4 = "100,";   Name4 = "E_lights";  }
```

Note that `E_tot` actually refers to the *cumulative days of simulation.* This is a work-around since EnergyPlus cannot compute equation (1). Hence, we will compute (1) directly in GenOpt by adding one line of code after reading the function values. To be able to report all values, namely $E_{tot}(x)$, $Q_{heat}(x)$, $Q_{cool}(x)$, and $E_{lights}(x)$, we need to read in *any* value for $E_{tot}(x)$ from the simulation output file. This will assign a variable in GenOpt for $E_{tot}(x)$ that we can then overwrite using equation (1). We stress that modifying GenOpt's code is usually not required, since most simulation programs allow the computation of (1) directly in the simulation program. Hence, with most programs, no code modification of GenOpt would be necessary. Rather than modifying GenOpt's source code, we could also have used some post-processing program to compute (1) or any econometrics calculation, e.g., to compute operating costs.

4.5 File Location

As the last step, we have to specify in the **optimization configuration file** where all the files in Figure 2 are located. After doing that, we can start the optimization.

5 Result of Optimization

To solve the optimization problem, we used the Hooke-Jeeves pattern search algorithm [HJ61], with improvements by Smith [Smi69], Bell and Pike [BP66], and De Vogelaere [DV68].

Using this algorithm, the total source energy consumption, $E_{tot}(x)$, was reduced by 14% compared to the initial values. Table 2 shows the initial values and the values corresponding to the minimizer of (1), and Figure 3 shows the values at each iteration step. To achieve the minimum point, 117 iterations were required. Since the algorithm required the objective function value at some points more than once (GenOpt detects such cases and, hence, does not perform a simulation run), a total of 98 EnergyPlus simulations were required. This took 4 hours on a 200MHz Pentium computer running Windows NT 4.

As shown in Table 2, the optimal building design has been achieved by increasing the window width and by rotating the building by 90° (the window initially facing west is now facing north). Even though the larger window width leads to slightly higher cooling energy consumption (due to increased solar gains), the electricity consumption for lighting was reduced by 19%. Note that this finding strongly depends on the fact that the building has an exterior shading device and a window overhang. Both measures reduce solar heat gains during summer. Also, due to increased solar gains, heating energy consumption has been reduced by 28%.

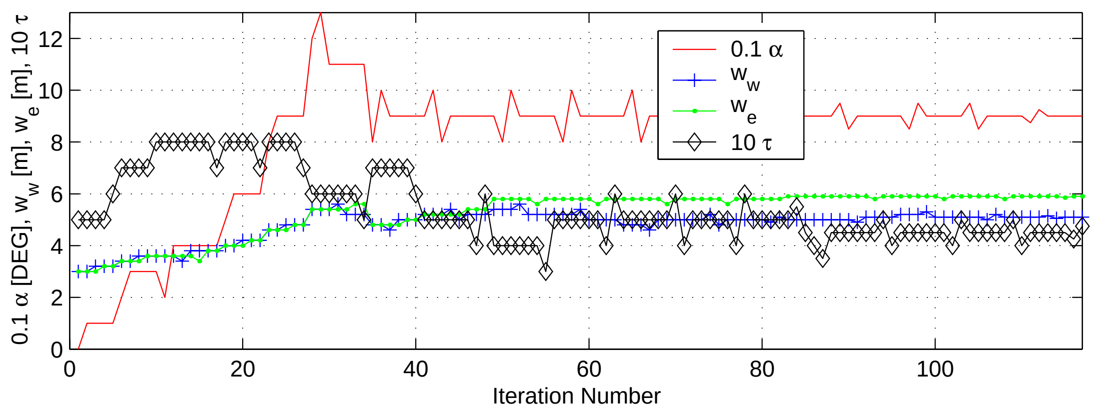
As can be seen from Figure 3, increasing τ in the 5th iteration reduced $E_{tot}(x)$ (otherwise, the algorithm would not have further increased τ). Hence, at the original building configuration, an increase in the shading device transmittance reduces $E_{tot}(x)$. This may be explained by the west-east window orientation, which causes large solar gains since, at sunrise and sunset, the window overhang is not very effective and the solar incidence angle is small. However, as the building is rotated by 90°, τ reduces to an optimal value of 0.45.

To show that the Hooke-Jeeves algorithm really converged to a minimizer, we also did parametric runs on 2400 possible parameter configurations, and an optimization using another algorithm[4]. It

[4]The simplex method of Nelder and Mead [NM65] with an extension of O'Neill [O'N71]

	initial value	optimum value
Source energy consumption E_{tot} [kWh/(m^2 a)]	168.1 100%	145.3 86%
Heating energy consumption Q_{heat} [kWh/(m^2 a)]	9.86 100%	7.14 72%
Cooling energy consumption Q_{cool} [kWh/(m^2 a)]	32.0 100%	34.5 108%
Lighting energy consumption E_{lights} [kWh/(m^2 a)]	34.7 100%	28.0 81%
building azimuth α [DEG]	0	90
width west window w_w [m]	3	5.1
width east window w_e [m]	3	5.9
shading device transmittance τ	0.5	0.45

Table 2: Initial and final values of optimization



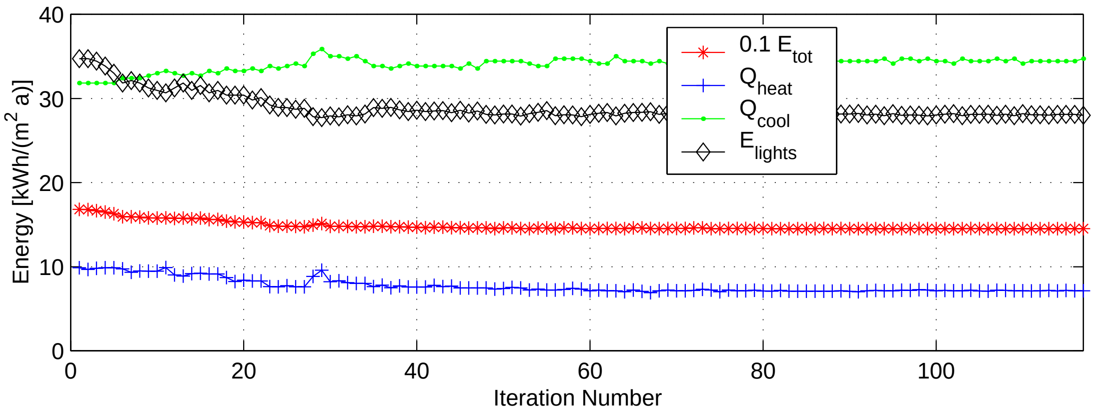
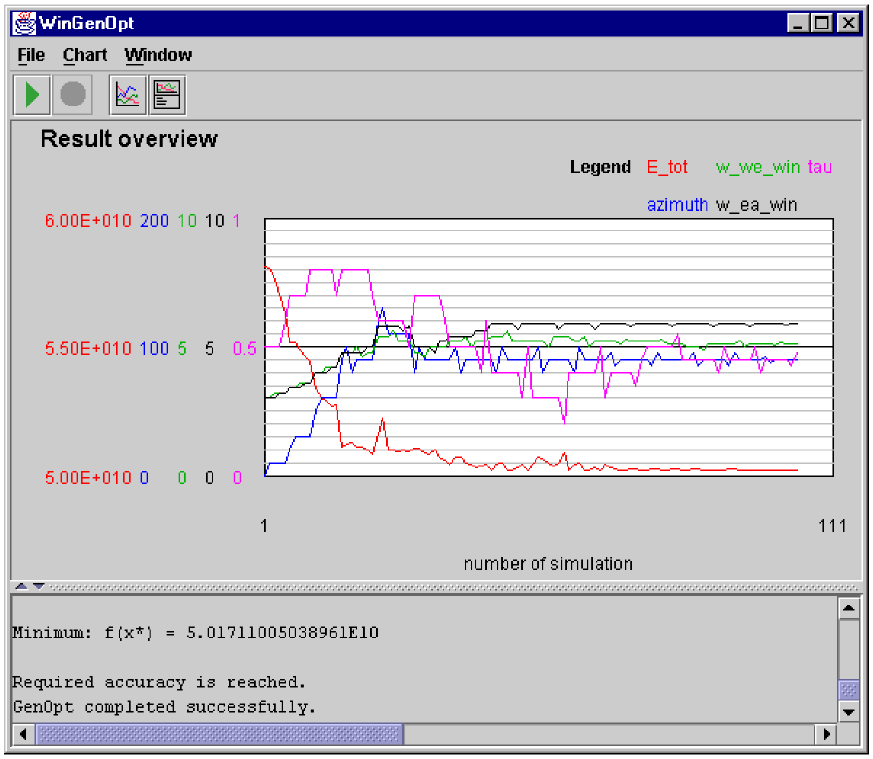


Figure 3: Iteration sequence of optimization algorithm



```
Minimum: f(x*) = 5.01711005038961E10

Required accuracy is reached.
GenOpt completed successfully.
```

Figure 4: GenOpt's display of the progress of an optimization run

turned out that both optimization algorithms converged to the same point. Also, the parametric runs did not reveal any parameter combination with a lower function value. Therefore, the solution obtained with the optimization is very likely the global minimizer.

In summary, the optimization leads to 14% lower source energy consumption (and hence lower energy costs) and better daylighting usage.

6 Availability

GenOpt 1.1 is expected to be released in November 2000. Like GenOpt 1.0, it will be downloadable free of charge from `http:\\SimulationResearch.lbl.gov`. The user manual can also be downloaded from this web site. Since GenOpt is written entirely in Java 1.2, it can be run on any operating system that supports Java (e.g., Windows NT/95/98/2000, Unix, Linux). The installation also contains simulation configuration files for different building simulation programs, such as EnergyPlus, DOE-2, SPARK and TRNSYS.

GenOpt can either be run as a console application (for example, from a batch job or shell script to do multiple optimization runs), or with a graphical user interface that shows the progress of the optimization (Figure 4). GenOpt 1.1 has four different optimization algorithms, two for multidimensional optimization (Hooke-Jeeves algorithm [HJ61] and simplex method of Nelder and Mead

[NM65] with an extension of O'Neill [O'N71]) as well as two line-search methods (Golden Section and Fibonacci interval division). The line-search methods allow the optimization of a function with respect to one independent parameter in a unimodal interval.

7 Conclusion

Since GenOpt is a stand-alone program that is configured for the used simulation program by text files only, it can be used to do optimization with *any* simulation program that has text-based input and output. Also, since GenOpt processes whatever ASCII input file it is given, the system being optimized can be far more complex than the one shown here. Hence, it is also possible to optimize building envelope design together with HVAC system design as long as the independent parameters are continuous.

If the number of independent parameters exceeds two, the designer usually cannot understand and quantify the nonlinear interactions of the various system parameters. For such cases, mathematical optimization is a cost-effective tool that supports the analyst in designing better systems, which often results in lower operating costs and higher comfort for the building occupants. Doing optimization also gives the designer a better understanding of the system behavior. The time required for setting up an optimization problem is usually less than one hour.

8 Acknowledgements

This work was supported by the US Department of Energy (DOE), the Swiss Academy of Engineering Sciences (SATW), the Swiss National Energy Fund (NEFF), and the Swiss National Science Foundation (SNSF). The author would like to thank these institutions for their generous support.

References

[BP66] M. Bell and M.C. Pike. Remark on algorithm 178. *Comm. ACM*, 9:685–686, September 1966.

[DV68] R. De Vogelaere. Remark on algorithm 178. *Comm. ACM*, 11:498, Jul. 1968.

[HF99] Joe Huang and Ellen Franconi. Commercial heating and cooling loads component analysis. Technical Report LBL-37208, Lawrence Berkeley National Laboratory, EETD, November 1999.

[HJ61] R. Hooke and T.A. Jeeves. 'Direct search' solution of numerical and statistical problems. *J. Assoc. Comp. Mach.*, 8(2):212–229, April 1961.

[NM65] J.A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, January 1965.

[O'N71] R. O'Neill. Algorithm AS 47-function minimization using a simplex procedure. *Appl. Stat. 20*, 20:338–345, 1971.

[Smi69] Lyle B. Smith. Remark on algorithm 178. *Comm. ACM*, 12:638, November 1969.