A Modified Genetic Optimization Algorithm Using Ancestor Path Extrapolation

Aaron Powers
Johnson Controls, Inc., Memphis, TN

Abstract

Optimization techniques have become increasingly popular in building performance simulation and design. The relatively high computational expense of simulating a building for a full year has traditionally been a major obstacle in the widespread use of these techniques. Consequently, most practical implementations employ some variety of meta-heuristics which do not necessarily guarantee an optimal solution but also do not require an impractical number of simulations to run. Still, further gains in convergence speed are needed for automated optimization to become truly practical in building design. This paper introduces a modified genetic optimization algorithm which implements a new creation operator to improve the rate of convergence in building performance optimization. The performance of the new algorithm was compared to that of a traditional genetic algorithm using a test case building model with 26 real valued design variables over several optimization runs. The new algorithm displayed a faster rate of convergence in all tests. The improved convergence rate was independent of the initial population fitness and building location. These results show that genetic algorithms with ancestor path extrapolations can greatly reduce the computational time required in building performance optimization.

Introduction

The goal of optimization in building simulation is to determine the set of design parameters that optimize one or more objective functions. The essential difficulty in these optimizations is that the evaluation of the most popular objective function (building energy consumption) is very time consuming. This has led to the wide use of metaheuristic optimization algorithms, which do not guarantee an optimal solution but also do not require a large number of calls to the building simulation engine compared to other methods (Nguyen et al. 2014).

One of the most popular classes of meta-heuristic algorithms used in building performance optimization is the class of genetic algorithms (GA). These algorithms attempt to optimize building performance by mimicking natural selection over a number of generations of potential solutions. Implementations of the basic GA have been successfully used to solve a variety of building performance optimization problems (Palonen et al. 2009, Rahmani Asl et al. 2014, Xu et al. 2016). The ready availability of the basic GA in packages such as jMetal (Durillo and Nebro 2011) and Optimo (Rahmani Asl et al. 2015) have also helped increase the popularity of this algorithm in building design.

Implementations of a genetic optimization algorithm can still be very time consuming, with run times being reported as high as 23 hours to simulate 30 generations (Xu et al. 2016). As a result, additional techniques have been used to reduce this burden. One of these techniques is to reduce the initial population size or the total number of generations; however, this can result in the selection of sub-optimal solutions. Another technique, the use of surrogate models, has been shown to effectively reduce the computational burden associated with running GAs on building models (Magnier and Haghighat 2009, Tresidder et al. 2011, Aijazi and Glicksman 2016). Surrogate models are simplified models which are constructed to have behavior similar to the actual detailed building model. This technique reduces computation time because the surrogate model executes much faster than the detailed model when evaluating the objective function; however, there is some loss in accuracy when using the surrogate model.

This paper introduces a method for creating new individuals in a genetic algorithm based on the paths created by ancestors. When used in conjunction with traditional crossover and mutation, this method can reduce the time required to approach an optimal solution.

Basic Genetic Algorithm

Overview

The basic genetic algorithm for single objective optimization can be defined as shown in Figure 1.

```
set $n$, the initial population size
initialize population $P = \{\vec{x}_1, \vec{x}_2, \dots \vec{x}_n\}$
compute fitness $\{f_1, f_2, \dots f_n\}$ of $P$
for each generation
    set $m$, the number of children to create
    select individuals to undergo genetic operations
    create children $C = \{\vec{c}_1, \vec{c}_2, \dots \vec{c}_m\}$ using genetic operators
    compute fitness $\{f_1, f_2, \dots f_m\}$ of $C$
    select individuals $S \subseteq P$ to survive to next generation
    set $P = C \cup S$
next generation
```

Figure 1: Basic Genetic Algorithm

Population Initialization

Each individual, $\vec{x}_i \in P$, is defined as a vector of genes that represent specific design parameters. These genes can represent parameters such as window properties, HVAC control points, and building geometry characteristics. Given a set of genes over which to optimize the performance of a building, initial individuals are created randomly over a predefined range.

Fitness Evaluation

The fitness of an individual in building simulation can be defined in many different ways. For the purposes of this paper, the fitness is defined as the annual source energy consumption of the building. Therefore, individuals that represent more energy efficient building designs will have better fitness values.

Selection for Operations

There are several options available for selecting individuals to undergo genetic operations in GAs. Binary tournament selection is the mechanism used in this paper. This mechanism selects an individual by returning the more fit of two randomly selected individuals from the population with a probability of 90%. That is, the lesser fit individual is returned 10% of the time. This feature allows for a more diverse gene pool and can avoid convergence to local optima.

Genetic Operators

The most common genetic operators in GAs are crossover and mutation. These operators attempt to mimic their counterparts observed in nature. Simulated binary crossover (SBX) is a very popular operator for real-coded GAs and is used as the crossover operator in this paper (Deb et al. 1995). The genes of the children individuals created through crossover are then altered with a mutation operator with a given probability. Polynomial mutation is used as the mutation operator in this paper (Deb et al. 1999).

Selection of Individuals for Survival

An optional feature of GAs is the ability to allow more fit individuals to survive for multiple generations. This feature is an example of a strategy known as elitism. In this paper, the most fit individual in each population is allowed to survive to the next generation.

Ancestor Path Extrapolation

Definition

Ancestor path extrapolation is an individual creation method that works in conjunction with the other creation operators. A k^{th}order ancestor path extrapolation creates a new individual $\vec{x}^{k+1}$ based on the sequence of k of its direct ancestors $\{\vec{x}^1, \vec{x}^2, \ldots, \vec{x}^k\}$. Superscripts are used to denote parental relationships. For example, $\vec{x}^3$ is the child of $\vec{x}^2$ and the grandchild of $\vec{x}^1$. A new individual is created using the following vector function

$$\vec{x}^{k+1} = \vec{x}^1 + \Delta\vec{x}^1\alpha + \frac{\Delta_2\vec{x}^1}{2!}\alpha(\alpha-1) + \cdots + \frac{\Delta_k\vec{x}^1}{k!}\alpha(\alpha-1)\ldots(\alpha-k-1) \tag{1}$$

where

$$\alpha = k + \lambda - 1 \tag{2}$$

$$\Delta\vec{x}^i = \vec{x}^{i+1} - \vec{x}^i \tag{3}$$

$$\Delta_j\vec{x}^i = \Delta_{j-1}\vec{x}^{i+1} - \Delta_{j-1}\vec{x}^i \tag{4}$$

and λ is a scaling factor which sets how far along the extrapolated curve to step.

Selection of k and λ

The behavior of the ancestor path operator can be greatly affected by the selection of k and λ. Figure 2 shows an example of four different orders of ancestor path extrapolations applied to a single-gene individual that has four direct ancestors. As the plot shows, when $(k-1) < \lambda < 1$ the operator results in an interpolation between the ancestors. In all cases, when $\lambda = 1$ the operator returns a copy of the parent. For a given value of λ, a higher order extrapolation generally (but not always) results in a larger change in the gene. Also, the direction of the extrapolation can change depending on the selection of k and λ.

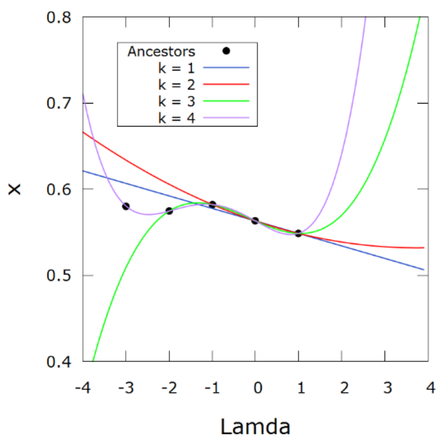

Figure 2: Effects of k and λ on a single-gene individual

Optimizations which favor lower values for k tend to display better convergence. For this paper, the value of k was set according to a probability distribution defined by

$$P(k=1) = 0.80$$
$$P(k=2) = 0.15$$
$$P(k=3) = 0.03 \qquad (5)$$
$$P(k=4) = 0.02$$

The value for λ was set according to

$$\lambda = \frac{2u}{k^2} \qquad (6)$$

where u is a random number between 0 and 1.

The $k = 1$ Case

The $k = 1$ case has several important properties. First, the direction of the extrapolation with respect to the parent will not change for $\lambda > 1$, and the Euclidean distance to the parent will increase as $\lambda \to \infty$. Second, in cases where $f^{i+1} - f^{i} < 0$, the extrapolation is guaranteed to result in reducing the objective function provided that $\vec{x}^{i+1}$ and $\vec{x}^{i}$ are relatively close (i.e. $\vec{x}^{i+1} - \vec{x}^{i}$ does not intersect local extrema or saddle points) and λ is small enough. Of course, there is a direct tradeoff between this advantage and the rate of convergence.

Parent Selection

The following procedure is used to select individuals to undergo ancestor path extrapolation in each generation. First, the change in fitness relative to the most similar parent is calculated for each individual in the population.

$$\delta = f^{i+1} - f^{i} \qquad (2)$$

Next, the subset of individuals for which $\delta < 0$ is sorted from the lowest values of δ to the highest. Finally, for n ancestor path extrapolations, the first n individuals in this sorted list are selected as parents.

Test Case

Building Description

A test building with characteristics shown in Table 1 was used to compare the performance of a basic genetic algorithm (GA1) to that of a genetic algorithm using ancestor path extrapolation (GA2). The test building was modelled and simulated using the DOE2.2 simulation engine. The building has a rectangular footprint with zoning as sown in Figure 2. Twenty-six optimization variables were selected including building geometry parameters, envelope properties, and control setpoints. The full list of optimization variables is shown in Table 2.

Table 1. Test building description

Item	Value
Location	Memphis, TN
Primary Usage	Office
Operating Hours	6AM – 6 PM Monday – Friday
Conditioned Area	80,000 ft^2
Floors	3
Lighting	1.0 Peak W/ft^2 3 Level Daylight Control
Equipment/Plug Loads	0.75 Peak W/ft^2
Air-Side HVAC	VAV with HW Reheat
CHW Distribution	Primary only variable flow with constant DP setpoint
Chillers	Water-cooled centrifugal with VSD controlled to constant CHW setpoint.
CW Distribution	Constant volume
Cooling Towers	Open towers with VSD controlled to constant CW setpoint.
HHW Distribution	Primary only variable flow with constant DP setpoint
Boilers	Natural-draft, natural gas fired.

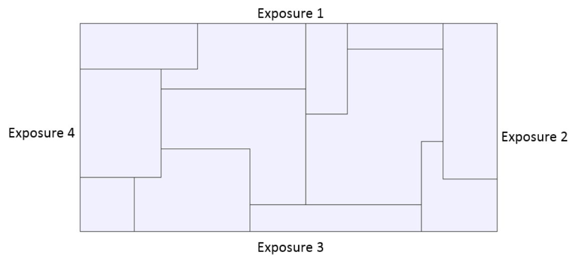

Figure 2. Test building zoning

Table 2. Optimization variables

Variable	Units	Min	Max
Building Azimuth	Deg.	-90	90
CHW Supply Setpoint	◦F	40	48
Cond. Water Setpoint	◦F	68	85
HHW Supply Setpoint	◦F	150	190

Variable	Units	Min	Max
Roof Absorptivity	-	0.1	0.9
Roof Emissivity	-	0.1	0.9
Per Exposure Variables (4 of each)			
Percent Glazing	-	0.05	0.80
Glazing U-Value	-	0.1	0.9
Glazing Shading Coefficient	-	0.1	0.9
Glazing Emissivity	-	0.1	0.9
Window Overhang Depth	ft.	0.0	8.0

Algorithm Descriptions

The baseline GA (GA1) follows the description given above with SBX and polynomial mutation as the creation operators. The parameters for this algorithm are shown in Table 3.

Table 3. GA1 parameters

Item	Value
Initial Population Size	30
Children Created per Generation	30
Mutation Probability	10%
Crossover Distribution Index	20
Mutation Distribution Index	20
Total Generations	100

The GA with ancestor path extrapolation (GA2) uses the same structure as GA1 with the exception that a certain number of children are created using ancestor path extrapolation rather than through crossover/mutation in each generation. In both GA1 and GA2, the number of calls to the DOE2 engine per generation are the same. The parameters for GA2 are shown in Table 4.

Table 4. GA2 parameters

Item	Value
Initial Population Size	30
Children Created per Generation	30
Max Ancestor Path Extrapolations per Generation	10
Mutation Probability	10%
Crossover Distribution Index	20
Mutation Distribution Index	20
Total Generations	100

In both cases, 100 generations was selected as the termination condition. This value was chosen because most GA2 runs were approximately converged by this generation.

Results

Five optimizations were run using both GA1 and GA2. Both algorithms were initiated from the same initial population for a given optimization run. A plot of the most-fit individual by generation is shown in Figures 3-7. The plots show that GA2 was able to converge at a faster rate than GA1 in all cases. For example, in the first optimization run, GA2 was able to eclipse the 100th generation best fitness of GA1 after only 34 generations. Similar results can be seen in the other optimization runs. The improved rate of convergence was independent of the fitness of the initial population.

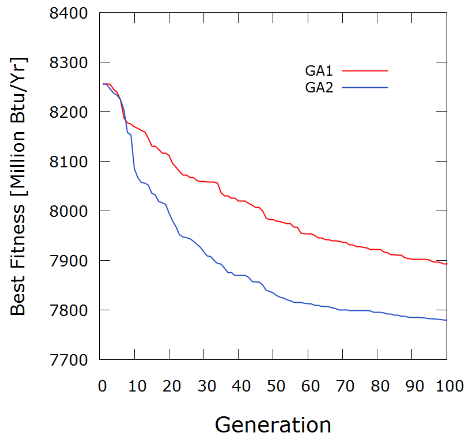

Figure 3. Optimization run 1

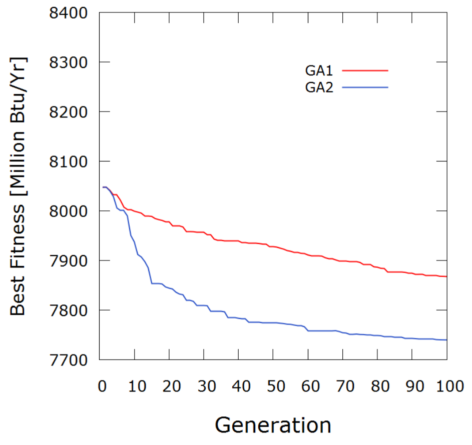

Figure 4. Optimization run 2

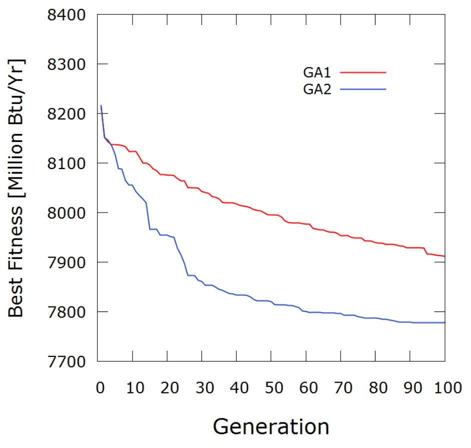

Figure 5. Optimization run 3

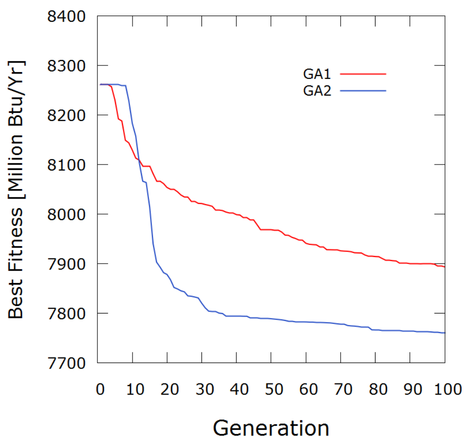

Figure 6. Optimization run 4

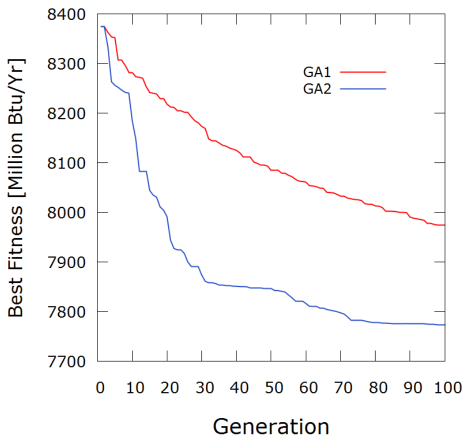

Figure 7. Optimization run 5

Further analysis of the results reveal a particular advantage of the ancestor path extrapolation operator. Table 5 shows the percent of individuals in the first optimization run that displayed an improved fitness compared to its parent for each creation operator. All orders of ancestor path extrapolation produced improved fitness children at a higher percentage than traditional crossover/mutation. This higher percentage of improvement leads to a faster convergence.

Table 5. Creation operator statistics

Creation Operator	**Count**	**Pct. Showing Improved Fitness**
SBX/Polynomial Mutation	2000	38.3
Ancestor Path $k = 1$	743	57.0
Ancestor Path $k = 2$	165	55.8
Ancestor Path $k = 3$	33	59.4
Ancestor Path $k = 4$	21	50.0

The new algorithm was also tested for other locations. Figures 8 and 9 show the results for optimizations of the test building in San Francisco and Chicago. The results show that GA2 also displays a faster rate of convergence for these locations.

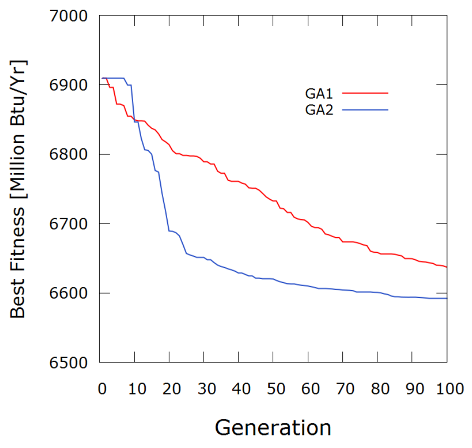

Figure 8. Optimization run for San Francisco

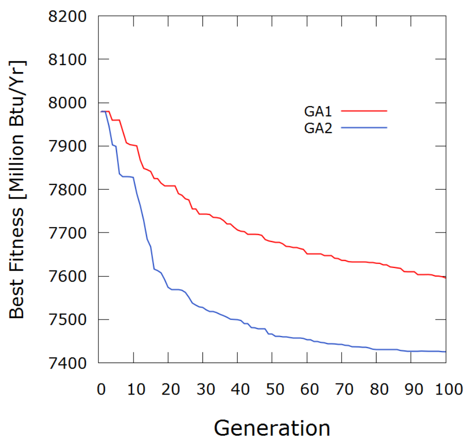

Figure 9. Optimization run for Chicago

Conclusion

Optimizations using genetic algorithms can be a useful tool in building design; however, the rate of convergence can be too slow to be practically useful for many practitioners. This paper introduced a new creation operator which uses extrapolated paths of an individual's ancestors to create offspring. Simulation results showed that this new operator can greatly increase the rate of convergence of genetic algorithms regardless of initial population fitness or building location. The use of the newly presented operator in building performance simulation can greatly reduce the computational time required for genetic algorithms to converge.

References

Aijazi, A.N., Glicksman, L.R. (2016) Comparison of regression techniques for surrogate models of building energy performance. *ASHRAE/IBPSA Simbuild 2016.*

Deb, K & Agarwal, R.B. (1999) A niched-penalty approach for constraint handling in genetic algorithms. *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA-99), 235-243.*

Deb, K & Agarwal, R.B. (1995) Simulated binary crossoverfor continuous search space. *Complex Systems,* 9, 115 – 148.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEE Transactions on Evolutionary Computation*, 6(2), 182-197.

Durillo, J.J., & Nebro, A.J. (2011) jMetal: a java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10), 760-771.

Nguyen, A.T., Reiter, S., & Rigo, P. (2014) A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, *113*, 1043-1058.

Magnier, L., & Haghighat, F. (2010) Multiobjective optimization of building design using TRNSYS simulations, genetic algorithm, and artificial nueral network. *Building and Environment*, 45(3), 739-746.

Palonen, M., Hasan, A., & Siren, K. (2009) A genetic algorithm for optimization of building envelope and HVAC system parameters. *Building Simulation 2009: Eleventh International IBPSA Conference,* 159-166.

Rahmani Asl, M., Bergin, B., Menter, A., & Yan, W. (2014) BIM-based parametric building energy performance multi-objective optimization. *Conference on Education and Research in Computer Aided Architectural Design in Europe*.

Rahmani Asl, M., Stoupine, A., Zarrinmehr, S., Yan, W. (2015) Optimo: A BIM-based multi-objective optimization tool utilizing visual programming for high performance building design. *33rd Annual Conference of eCAADe*, 673-682.

Tresidder, E., Zhang, Y., & Forrester, A.I.J. (2011) Optimisation of low-energy building design using surrogate models, *Building Simulation 2011: 12th Conference of IBPSA*, 1012-1016.

Xu, W., Lam, K.P., Chong, A., & Karaguzel, O.T. (2016) Multi-objective optimization of building envelope, lighting, and HVAC systems designs. *ASHRAE/IBPSA Simbuild 2016.*