COUPLING TRNSYS AND MATLAB FOR GENETIC ALGORITHM OPTIMIZATION IN SUSTAINABLE BUILDING DESIGN

Marcus Jones
Vienna University of Technology, Vienna, Austria

ABSTRACT

Incorporating energy efficient features into sustainable buildings is cost effective during the design phase. In designing a sustainable building, the designer is faced with a staggering amount of parameters, conditions, and objectives. However, costs and limited time do not allow the designer to effectively evaluate all candidate building designs. Genetic algorithms are interesting for this problem given their ability to optimize complex multi-objective non-smooth optimization problems. The application of genetic algorithms to the problem of sustainable building design has been developed and will be presented. A method of interfacing TRNSYS and the Matlab genetic algorithm toolbox has been tested by application to two simple energy design problems. The approach of coupling the detailed modeling capabilities of TRNSYS and genetic algorithm routines in Matlab is powerful combination in the search for optimal sustainable building designs.

MOTIVATION

Sustainable building certification programs are becoming ever more popular, with their principles moving from voluntary rating levels into legislated requirements. The motivation is clear, sustainable building construction, operation, and decommissioning will play an important role in a sustainable future. However, sustainable building design is a highly complex process.

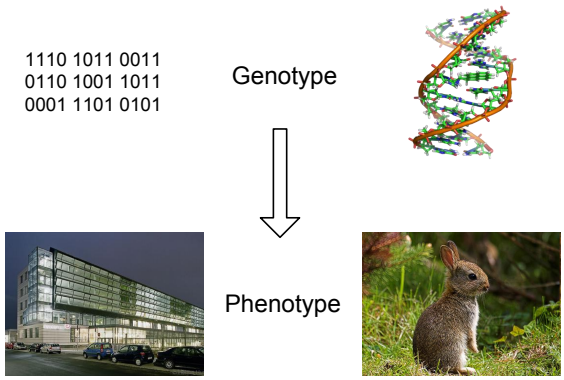

Figure 1: Building design and evolution

The conditions in which the building is designed and operated should be well understood. These are boundaries which the designer has little or no control over, and include the weather, surroundings, availability of energy resources, economic considerations, and building loads. The second aspect is the design of the building and system defined by the parameters which the designer does have control over. These include the parameters of the building envelope shape and materials, the energy system components and how the system is controlled and operated. Finally, the design objectives should be clear. Specific objectives include comfort levels, operating and investment costs, carbon reduction, and aesthetics.

In mathematical terms, this is a multi-objective optimization problem, and can be formalized as;

$$\min_{\vec{x}} [\mu_1(\vec{x}), \mu_1(\vec{x}), \ldots, \mu_n(\vec{x})]$$

where $\vec{x}$ is a vector in design space subject to the design constraints (a candidate solution), and μ_i is the i-th objective function, a component of the $\vec{\mu}$ solution vector. The solution to the problem is the set of Pareto optimum (Bui and Alam 2008). In optimization problems, the design variables define a solution space which is methodically searched in order to find the best candidate solutions. Building design problems are complex, often non-smooth in the solution space, and multi-objective. Because of the interacting nature of the design variables, unintuitive solutions exist which a designer may not consider. Because of the number of solutions, the solution space must be efficiently searched. Approaches to finding good solutions include past experience, trial and error, simple calculations, and building simulation. Because of the number of design variables and their interacting and dynamic nature, building simulation is becoming a more popular tool in the construction industry, becoming integrated within the architectural process.

For this type of optimization problem, genetic algorithms provide a robust and efficient optimization method (Wetter 2004). Genetic algorithms are parallelizable, so many function evaluations can be distributed across multiple processors (Eiben and Smith 2007). Because genetic algorithms have an element of random search, unintuitive but very effective solutions can be found (Keane and Brown 1996).

GENETIC ALGORITHMS

In nature, evolution serves as the mechanism in searching for the best candidate solutions to the complex prob-

lem of survival. In abstract terms, an organism can be defined by DNA. Individuals which are best adapted to the environment propagate their DNA to further generations. Small random variations create new traits, to increase diversity in the population. This mechanism has produced the incredible variety of organisms in nature, excellent solutions to the highly complicated problem of survival. This same mechanism can be applied to sustainable building design.

Figure 2 shows a generic genetic algorithm. First, an initial population is defined, usually containing randomly generated individuals. Each individual in the population is then evaluated. In the case of building simulation, this is the execution of the simulation. The number of evaluations and the average time required for one evaluation typically defines the time required for the overall optimization run. The results of the evaluations are then assigned a fitness level, which represents the individual's proximity to the optimum. Individuals which best meet the objectives of the problem are then selected to propagate the next generation, to form the parent population. The parents then recombine to form the unmutated population. Random variation is applied to this population to produce the next generation. This cycle repeats until a stopping condition is reached a fixed number of generations or a number of generations in which successive iterations do not produce significant improvement.

Successful design of a genetic algorithm involves minimizing the time required to find the optimum solution. A successful algorithm could depart significantly from the generic algorithm described above depending on the application, with many parameters and strategies to be considered.

INTERFACE TO SIMULATION

The simulation tool under consideration is TRNSYS, v.17 (Solar Energy Laboratory, Univ. of Wisconsin-Madison 2009). The optimization will be performed within Matlab 2009, using the Genetic Algorithm Toolbox (The Mathworks 2009). TRNSYS is a dynamic simulation tool developed over 30 years, with flexibility in modeling systems and buildings. In TRNSYS, a model is developed within the Simulation Studio environment a graphical user modeling interface. The model is composed of 'Types', predefined components and algorithms which model the behavior of common systems. The model is saved in ASCII text format within in a 'deck' file (.DCK), which stores the Types and connections between them. The model is executed by TRNEXE, an algebraic and differential equation solver which iteratively computes the system state at each time step. Because the model is stored in text format, the model can be parameterized by a scripting languages such as Matlab.

Figure 3 shows a method of scripting the parameterization of the TRNSYS input .DCK file. First, the modeler creates the overall system model within the TRNSYS Studio, being careful to parameterize the model using text variables and defining proper output files. A method of standardizing TRNSYS simulations has been described in a previous publication (Jones and Ledinger 2010). The modeler then saves the model to the .DCK file. Within Matlab, the .DCK is copied so as not to alter the original file. Parameters such as the weather file, the timestep, and convergence tolerances are then updated by searching and replacing the relevant sections of the current .DCK file. Next, Matlab searches and replaces the values for any design variables. The modeler should define the design variables in a standard and clear manner such as `GeneticVar1 = 10, GeneticVar2 = 0.75`. The updated .DCK file is then executed by TRNEXE using the `dos()` command in Matlab, which is a simple command-line access function. The result files are parsed and loaded into Matlab. Results are then post-processed to produce the final objective variables. With this method, a powerful way of executing a model becomes available within Matlab. For example, by simply executing the command;

```
Energy = evalTRNSYS([4 45 200], Vienna, 5)
```

the annual electrical energy demand of a solar domestic hot water system is returned. In this case, the system has a design vector defined by a collector area of 4 m^2 at 45° to the horizontal, with a primary hot water storage tank volume of 200 L, and parameterized using the weather file for Vienna at a 5 minute time step. Wrapping the execution of TRNSYS into a Matlab function provides the capability to evaluate thousands of candidate solutions with no user quickly and easily.

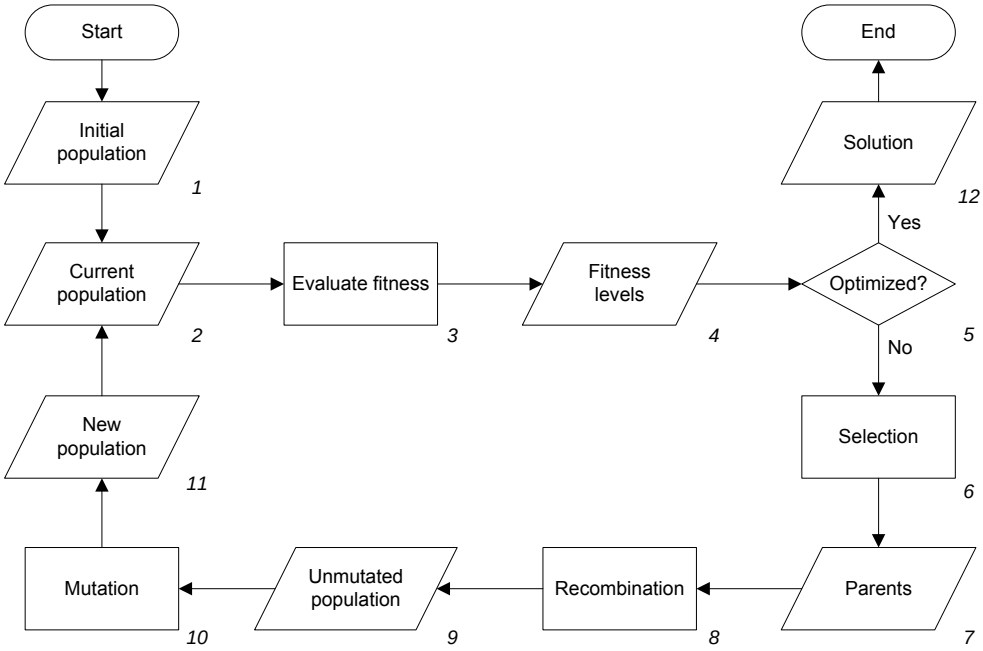

Figure 2: A generic genetic algorithm

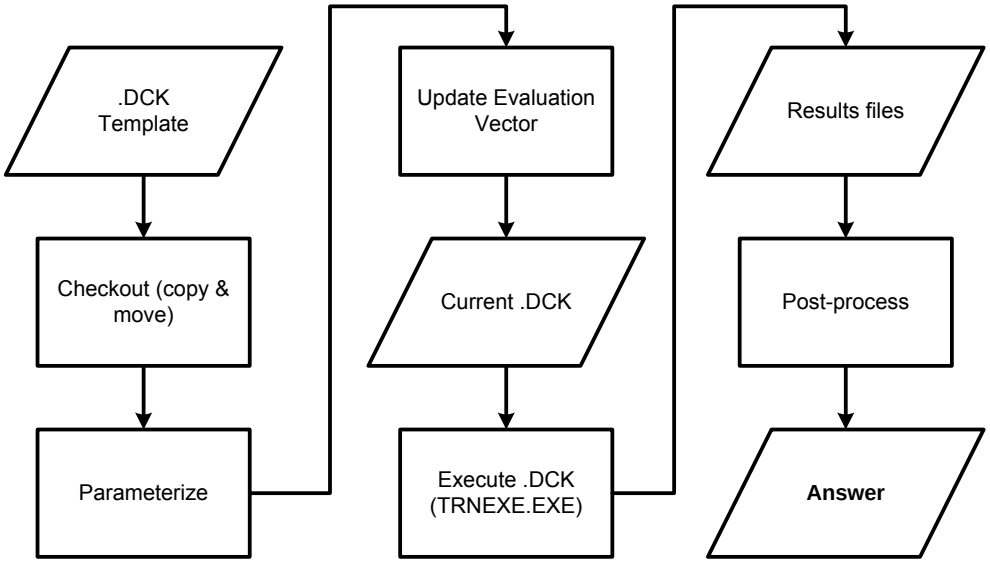

Figure 3: MATLAB - TRNSYS function wrapping

APPLICATIONS

Two problems were modeled, parameterized, and interfaced to the Matlab Genetic Algorithm Toolbox.

Simple building

The first model is based on the geometry of "Case 600" of ASHRAE standard 140-2007, a standard defining test cases for building energy simulation programs (ASHRAE 2007). The simple "shoebox" building seen in figure 4 is optimized with the objective of minimizing the annual heating energy demand for Vienna weather conditions. The design variables, summarized in table 1, are the building orientation, and the glazing area. A global search of the complete domain was performed, shown in figure 5. The first observation is that the design space is symmetric along the 180 °line, reducing the size of the search area by half. The glazing area influences the optimum orientation. If glazing is minimal (no windows), the orientation does not influence the optimum orientation. This is because no solar gains are possible to offset the heating demand. If glazing is maximized, the south orientation becomes the optimal, benefiting from solar heat gains through the windows, and the north orientation is the least optimal. Therefore, the optimum solution to this design problem is $x_1 = 0°$ (south) and $x_2 = 6\,\mathrm{m}^2$, with a value of $3549.0\,\mathrm{kWh/a}$ heating energy (per annum).

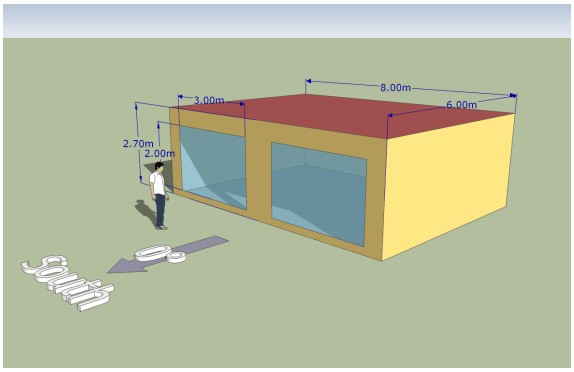

Figure 4: Simple building geometry

Table 1: Design variables for simple building

Variable	Range	Units	Description
x_1	$0 \leq x_1 \leq 180$	°	Orientation
x_2	$0 \leq x_2 \leq 6$	m^2	Glazing area

This problem was modeled in TRNSYS and coupled to the Matlab genetic algorithm toolbox using the method described above. The default genetic algorithm toolbox

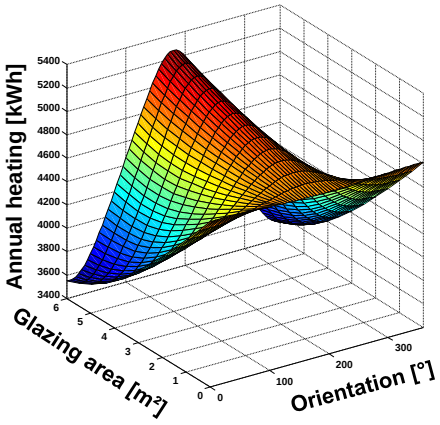

Figure 5: Global search of design space

parameters, listed in table 2, were used.

Table 2: Matlab GA default parameters

Initial population	-	20
Representation	-	Double
Fitness scaling	-	Rank
Selection	-	Stochastic
Elite count	-	2
Crossover	-	Scattered
Crossover fraction	-	0.8
Stopping	-	Tolerance / Generations
Tolerance value	-	0.5
Stall size	-	100
Mutation	-	Adaptive Feasible

A common way to display the progress of a genetic algorithm optimization is to show the number of generations on the x-axis, and show the value of the objective function on the y-axis. The best and worst solutions in the population at each generation are plotted, showing the range of solutions. The average population fitness is also plotted. Figure 6 shows the progress of the simple building design problem using the default algorithm. A common genetic algorithm profile is seen, with a fast increase in fitness in the first few generations, and smaller improvements throughout successive generations. These plots are useful for analyzing an important quality of genetic algorithms; diversity. The diversity of the population is represented by different metrics such as the spread of solutions or the average distance between candidate solutions in the design space. In figure 5, diversity quickly decreases as all individuals in the population converge to the optimum of $3549.0\,\mathrm{kWh/a}$. Managing diversity involves ensuring that diversity is high enough in order to properly search the design space, without being so high as to make the

number of function evaluations prohibitively high.

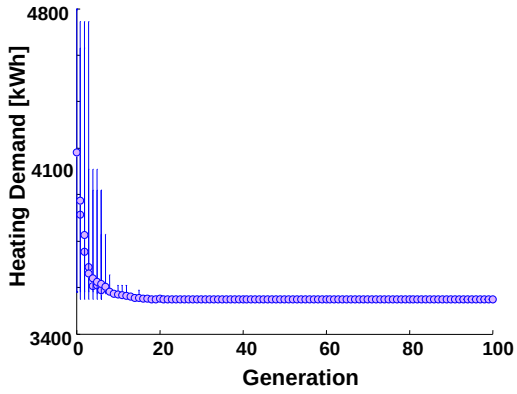

Figure 6: Genetic algorithm progress

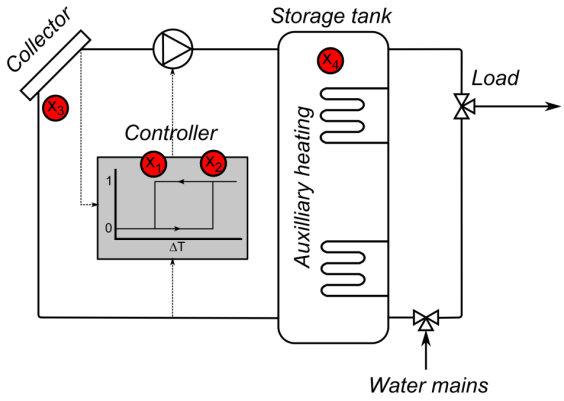

Figure 7: Simple solar domestic hot water system

Solar domestic hot water system

A second design problem is presented in figure 7, a simple solar domestic hot water system. The system consists of a collector field, a storage tank, and a differential controller with hysteresis. The design variables are presented in table 3. The objective is to minimize the amount of auxilliary electric resistive heating in the tank required to meet the hot water demands of a family dwelling in Vienna.

Table 3: Design variables for simple solar system

Variable	Range	Units	Description
x_1	$0 \leq x_1 \leq 30$	°C	Lower differential
x_2	$0 \leq x_2 \leq 30$	°C	Deadband
x_3	$0 \leq x_3 \leq 90$	°	Slope
x_4	$0 \leq x_4 \leq 0.500$	m^3	Volume

Because the design vector is four dimensional, a complete global search is computationally expensive. Instead, a manual searches was performed, first over the $[x_1, x_2, 45, 0.300]$ plane, producing an optimum of $1661.0\,kWh/a$ at $[0, 0, 45, 0.300]$. A second search over $[0, 0, x_3, x_4]$ further optimized the design to $1657\,kWh/a$ at $[0, 0, 45, 0.365]$. Therefore, the expected optimum design is to have a lower differential and hysteresis of $0\,°C$, a collector slope of 45°, and a tank volume of $0.365\,m^3$. From a practical viewpoint, the controller settings are not realistic since this causes pump cycling, but still provide a suitable test for the optimization routine.

A number of runs were executed, summarized in table 4. A simple investigation into the influence of population size on solution convergence was performed, with the first 5 runs having a population of 5 individuals, and the next 5 runs having a population of 20 individuals. In all runs 100 generations were evaluated. In the case of the population size of 20, better solutions are achieved, the best being $1655.1\,kWh/a$ at $[0, 0, 43.3, 0.369]$. With a population size of 5, the solutions are more varied, with an extreme case of $2489.7\,kWh/a$ at $[12.3, 1.5, 39.6, 0.279]$. Clearly, the initial random population of 5 individuals does not provide enough initial diversity to ensure an optimum solution. One strategy with a low population size is therefore to increase the mutation rate to increase diversity over the generations. With 20 individuals, the chances are better that a random initial individual is close to the optimum and able to propagate a better design throughout the generations. The cost of the larger population size is the potential for increased number of evaluations. In this case, the run time was simply the average simulation time of each individual multiplied by the population size, and multiplied by the number of generations which was fixed at 100. A stopping criteria based on convergence would produce a varying number of generations and a better indication of evaluation time cost.

CONCLUSIONS AND FUTURE WORK

Genetic algorithms are powerful tools in finding solutions to complex design problems. Their advantages include parallelization, non-intuitive optimum solutions, and a robust search regardless of the smoothness of the underlying function. Genetic algorithms are therefore interesting in the problem of sustainable building design. The simulation tool TRNSYS was coupled to the scripting platform Matlab with the Genetic Algorithm Toolbox. By wrapping the simulation execution in a function, the modeler can execute TRNSYS thousands of times in an automated fashion. Two simple case studies were used to show the feasibility of using genetic algorithms in optimizing the energy design of a simple building and a simple solar

Table 4: Run history for solar hot water system

Run	Generations	Population	Time [hr]	x1	x2	x3	x4	Best
1	100	5	3.3	2.1	0.6	44.1	0.419	1728.0
2	100	5	3.2	0.3	2.1	42.3	0.468	1665.5
3	100	5	3.0	0.0	9.3	44.1	0.351	1663.1
4	100	5	3.3	4.2	4.2	45.9	0.473	1864.2
5	100	5	3.6	12.3	1.5	39.6	0.279	2489.7
6	100	20	13.6	0.0	1.5	43.2	0.392	1657.1
7	100	20	13.7	0.0	0.0	43.3	0.369	1655.1
8	100	20	13.7	0.0	0.0	43.2	0.375	1656.2
9	100	20	14.0	0.0	0.7	43.4	0.360	1655.2
10	100	20	14.1	0.1	0.0	43.2	0.390	1655.2

domestic hot water system. In these case studies, the basic concept was proven to be successful. Managing diversity and number of evaluations were determined to be critical issues in designing a successful genetic algorithm for sustainable building design. In any optimization problem, the underlying design space should be physically well understood by the modeler in order to identify symmetries and to evaluate candidate solutions from a practical perspective.

This research will be continued to optimize the genetic algorithm for sustainable building design, with the objective of an optimization platform that is flexible enough to handle any type of design vector for a multi-objective sustainable building design problem. The parameters and method influencing population size and configuration, selection, mutation, and stopping criteria will be investigated to design a robust genetic algorithm. Such a platform well be beneficial in the design of the next generation of sustainable buildings.

ACKNOWLEDGEMENTS

This work is developed within the doctoral program at the Technical University of Vienna, Institute of Computer Technology under the supervision of Dr. Dietmar Dietrich, and with support from the Austrian Institute of Technology.

REFERENCES

ASHRAE. 2007. "Standard 140 - Standard Method of Test for the Evaluation of Building Energy Analysis Computer Programs." Technical Report, American Society of Heating, Refrigerating and Air-Conditioning Engineers.

Bui, Lam Thu, and Semeer Alam. 2008. *Multi-Objective Optimization in Computational Intelligence*. Information Science Reference, Hershey PA.

Eiben, A.E., and J.E. Smith. 2007. *Introduction to Evolutionary Computing*. Springer.

Jones, Marcus, and Stephan Ledinger. 2010. "Pushing the limits of simulation complexity - a building energy performance simulation of an exhibition centre in the U.A.E." *Proceedings of Simbuild 2010 - 4th national conference of IBPSA-USA (Pending final acceptance)*.

Keane, A.J., and S.M. Brown. 1996. "The design of a satellite boom with enhanced vibration performance using genetic algorithm techniques." *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control*.

Solar Energy Laboratory, Univ. of Wisconsin-Madison. 2009. *TRNSYS 17 - a TRaNsient SYstem Simulation program*.

The Mathworks. 2009. *Matlab R2009b*.

Wetter, Michael. 2004. "Simulation-Based Building Energy Optimization." Ph.D. diss., University of California, Berkeley.