Building Simulation 2009

Eleventh International IBPSA Conference
Glasgow, Scotland
July 27-30, 2009

A GENETIC ALGORITHM FOR OPTIMIZATION OF BUILDING ENVELOPE AND HVAC SYSTEM PARAMETERS

Matti Palonen, Ala Hasan, Kai Siren

HVAC Technology, Helsinki University of Technology, P.O Box 4400, FIN-02015 HUT, Finland

ABSTRACT

The aim of this paper is to describe the features of a Genetic Algorithm (GA) developed to solve simulation-based optimization problems for the optimal design of building parameters. This GA has been developed using guidelines from top researchs in the field of evolutionary computation. It is mostly based on NSGA-II and Omni-optimizer. It can be used for single and multi-objective optimization problems with and without constrains. Both discrete and continuous variables can be handled.

Real-world optimization problems in the field of building performance simulation are carried out to verify the performance of the developed GA. Results for a single-objective optimization problem are presented, where the aim is minimization of life cycle cost of a detached house. Besides diverse sets of non-dominated solutions results for a multi-objective building design problem are also presented.

INTRODUCTION

The use of simulation tools is increasing in the design and performance analysis of buildings. Simulation-based optimization can be implemented to find optimal values of selected design variables in the building and HVAC system design problem to minimize (or maximize) any selected objective function(s). Those design variables can be continuous variables (with lower and upper bounds), discrete variables, or both. Examples of these are insulation thickness, window type, envelope U-values, building orientation, HVAC system type, components of the HVAC system, etc. Examples of the objective function are investment cost, operating cost, annual energy consumption, CO2 emissions, indoor air quality, equipment efficiency etc.

The success of the optimization is strongly affected by the properties of the problem, formulation of the objective function and by the selection of an appropriate optimization algorithm.

A genetic algorithm (GA) is a search technique used in computing to find solutions to optimization problems. Genetic algorithms can be categorized as meta heuristics with global perspective. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. Genetic algorithms are implemented as a computer simulation in which a population of abstract representations of candidate solutions to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. Appendix A presents main features of implemented GA.

First objective of this study is to use NSGA-II [1] and Omni-optimizer [2] as framework in development of a genetic algorithm to solve constrained mixed integer (Nonlinear) programming (MINLP) problems. These kind of problems often arise in the field of building and HVAC design.

A detailed study of balance between diversity preservation and convergence properties of Omni-Optimizer is performed. Different evolutionary

strategies such as restricted mating, adaptive mutation, genetic rebirth and removing of the duplicate solutions are also implemented and studied.

Second objective of this study is to make the use of this algorithm user friendly. It will be shown that implemented GA can be used to solve single-objective, multi-objective and constrained multi-objective real-life optimization problems without need to change or optimize algorithms parmeters.

Third objective is to implement hybrid algoritm which first does global search using GA and then switches to Hooke-Jeeves algorithm.

Fourth objective is to study the effect of scaling the search space in single-objective problems. That is, the effect of reducing number of bits used to represent the continuous variables is going to be investigated. Also attention is paid to defining the stopping criterion for the GA and for the hybrid algorithm.

Finaly, and as a fifth objective, algorithm should have the three following features:

1. Return extreme values of all objective functions.
2. Return good approximation of the true Pareto-front.
3. Return diverse approximation of the true Pareto-front.

It will be shown that in real life multi-objective optimization problems of buildings and HVAC systems, all of above mentioned features are achieved with a single run of the implemented algorithm.

PROBLEM

In a recent study [3], minimization of the life cycle cost (LCC) of an electrically heated single family detached house was achieved using combined simulation and optimization. Fig. 1 shows the dimensions of the house. The coupled simulation tool (IDA ICE 3.0[4])) and optimization tool (GenOpt 2.0[5]) were used to find optimum values of the decision variables in the house envelope and the HVAC system.

Five variables were considered in that study: three continuous variables and two discrete variables. The continuous variables were additional insulation thickness of the existing insulation material in the structure (external wall, roof and floor). The discrete variables were the U-value of the window (with two options, 1.4 or 1.0 W/m2K), and type of heat recovery (with two options, plate type with 70% efficiency or rotary type with 80% efficiency).

The absolute value of the LCC is not calculated, but the difference dLCC between the LCC for any case and that for the reference case. This way, there is no need to include cost data for all components of the building and system but only the differences produced by the variation of specified parameters between the reference case and any other case.

$$dLCC = dIC + dOC$$

where dIC (€) is the difference in the investment cost for the construction (insulations and windows) and ventilation heat recovery, while dOC (€) is the difference in the operating cost due to the difference in electric energy consumption for space heating only discounted to the present value.

Since the objective function is the difference between the life cycle cost for any case and that for the reference case (dLCC), thus dLCC = 0 represents the reference case. In order to get a better design, the solution should produce negative dLCC values, i.e. values of LCC lower than that for the reference case. The task is to find optimized values of the specified design variables that result in a minimum dLCC value.

The algorithm used in the optimization in [3] was a hybrid global optimization algorithm that initially does a Particle Swarm optimization for the continuous and discrete variables and then switches to the Hooke-Jeeves Generalized Pattern Search algorithm to refine the continuous variables.

The results from that study indicates that for number of years of 20 and escalation in electric price of 1 %, the minimum value of the objective function achieved dLCC is -2102 €, which was obtained after 260 simulations.

In the current paper the developed Genetic Algorithm will be used to solve this single-objective problem. The obtained results will be compared with those available from the above mentioned study [3].

Then, the problem is changed into a two-objective problem. This problem is solved with implemented multi-objective approach and results are compared with brute force results.

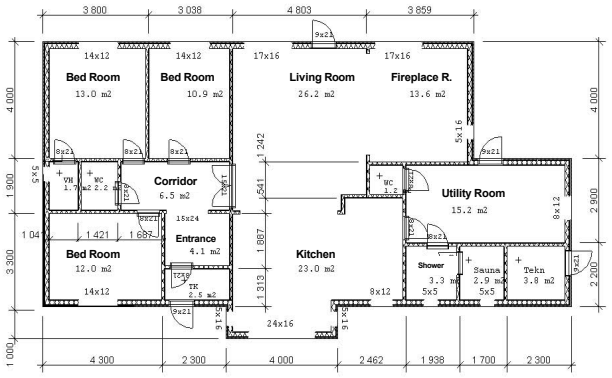

Figure 1 Dimensions of the detached house [3]

ALGORITHM

The Omni-optimization algorithm adapts itself to solve different kinds of optimization problems - single or multi-objective problems and uni- or multi-global problems [2]. Deb and Tiwari presented simulation results on various test problems, thus showing that Omni-optimizer is efficient handling problems with continuous variables. Real-parameter crossover and mutation operations used in their study are able to handle only continuous variables.

Use of the binary-coding in our implementation allows representation of both discrete and continuous variables. Gray-coding is also implemented to overcome the diffculties faced when using traditional binary-coding [6]. For problems with only continuous variables, a real parameter version is implemented with simulated binary crossover operator (SBX) and polynomial mutation as suggested in [1] and [2]. For binary-coded variables two-point crossover operator and bit-wise mutation operator are used.

In building simulation and optimization, problems having discrete and continuous variables are more usual than problems with only continuous variables. MINLP-problems are harder to solve than problems with only continuous variables. MINLP-problems can not be solved globally with algorithms that require continuous or differentiable search space. However, MINLP-problems can be divided into sub problems, which may have properties needed by other algorithms. GA usually convergences near optimal solution, but has difficulties obtaining the exact solution. This leads to the sub-problem, which could be solved with some local search algorithm.

To improve results returned by Omni-optimizer a external archive is implemented where all points visited by the algorithm are stored. This modification will be shown to improve results significantly. If for reasons related to computer memory, all points can not be recorded, a special case of this modification which needs no extra memory is to return best points from union of two last generations.

Our implementation starts with randomly initialized population P_0 of size N. After that population R_0 of size $2N$ is created from union of two random sampling of P_0. Finally selection is performed on R_0 by N binary tournament selection operators. One generation of implemented GA first constructs population R_t as above, here t is generation counter. Selection is performed with tournament selection operator to produce population Q_t from R_t if the restricted mating is not selected. Then the crossover and the mutation operators are applied to Q_t. Next population of size $2N$ is constructed from union of P_t and Q_t. This population is ranked with a user selected variation of the sorting procedure to produce fronts $F_1,F_2,\ldots$ Each member in front F_j is assigned crowding-distance value and fitness value equal to front number it belongs to. By default crowding-distace values are computed as defined in [2], but for advanced user other implementations are also available. Finally population P_{t+1} for next generation is created by selecting best N solutions or best N distinct solutions from ranked $2N$ solutions. After last generation, solution set is generated by ranking all solutions visited by the algorithm. Appendix A presents flowchart of implemented algorithm.

In all optimization runs presented in this paper, the selection operator proposed in [2] is used. Selection operator prefers feasible solution over infeasible solution. If both solutions are infeasible, one with smaller overall normalized constraint violation is selected. In case where both solutions are feasible, one which dominates other is selected. If both solutions are non-dominated, one with better crowding-distance value is selected.

Ranking procedure in omni-optimizer does not prefer feasible solution over infeasible. This fact makes Omni-optimizer non elitists approach in constrained problems if the initial population includes one or more infeasible solutions. In algorithm implemented a different ranking approach is prefered, where solutions are compared using constrained epsilon-domination principle. This modification makes algorithm more robust in constrained problems. Similar approach was used in NSGA-II, where comparisons were done using constrained domination principle.

Implemented procedure is expected to solve the multi-objective optimization problems in a manner similar to the NSGA-II with possible more diverse

solutions set in context of variable space. Detailed description about NSGA-II can be found in [1]. Our implementation is expected to return bigger set of solutions than NSGA-II in most multi-objective problems. Same argument stands for our implementation of Omni-optimizer. This argument follows from the fact that both NSGA-II and Omni-Optimizer might lose feasible solutions because of the constant population size *N*. Our implementation does not lose these solutions because of implemented external archive.

If the population size *N* is small, omni-optimizer might lose solutions to multi-modal problems. Because of external archive, our implementation does not have this property.

Every solution generated by the algorithm is evaluated exactly once. If some solution is generated several times there is no need to re-evaluate. This decreases simulations needed by the algorithm.

In the case of single-objective problems with continuous variables, procedure behaves similarly like standard single-objective evolutionary algorithms (EA) such as CHC or (μ+λ)-ES [2][6]. Moreover, with proposed crossover operator for continuous parameters, there is a remarkable similarity how self adaptive EAs and proposed algorithm work [6].

USE WITH GENOPT

GenOpt is an optimization program for the minimization of a cost function that is evaluated by an external simulation program. It has been developed for optimization problems where the cost function is computationally expensive and its derivatives are not available or may not even exist. GenOpt can be coupled to any simulation program that reads its input from text files and writes its output to text files [5].

Developed GA algorithm is planned to be added to GenOpt algorithm library. It extends the ability of GenOpt to tackle constrained multi-objective problems without need to define barrier or penalty functions.

When you have set up link with simulation programand GenOpt, defined decision variables and their bounds then you can easily use implemented GA with default setting by just defining probability of crossover and mutation, population size, number of generations and number of objective and constraint functions. This information is written to GenOpt command file as follows:

```
Algorithm {

Main = GA;
Objectives = 2;
Constraints = 1;
MutationProbability = 0.03;
CrossoverProbability = 0.8;
PopulationSize = 40;
Generations = 100;

}
```

This is the exact form that was used when the constrained multi-objective problem described later was solved.

Thus, minimum required number of parameters is four. Because of this and parameterless constraint handling strategy the use of the implemented algorithm is user friendly. Advanced users can tune the algorithm to 26 different evolutions strategies each having three different solution coding schemes.

OPTIMIZATION RESULTS FOR A SINGLE-OBJECTIVE PROBLEM

In this section, the optimization results for the single-objective problem are presented. The objective is to minimize dIC (€), the difference in the investment cost for the construction (insulations and windows) and ventilation heat recovery.

Parameters used for the each run of the algorithm are presented in a table 1. Crossover probablity of 0.8 was used in each run of the algorithm and mutation probabilty was always set to $1/\ell$, where ℓ is the length of the string used to represent the solution. Column Bits stands for the number of the bits used to represent the continuous variables. There were no attempt to optimize the parameters. Adaptive mutation, restricted mating, and the removing of the duplicate solutions are not active in runs presented in this and the following sections.

Table 1

Parameters used for each run of the GA

SINGLE-OBJECTIVE PROBLEM							
Crossover	Mutation	Population	Generations	Runs	Bits	ℓ	Coding
0.8	0.031	10	50	10	10	32	Binary
0.8	0.034	10	50	10	9	29	Binary
0.8	0.038	10	35	10	8	26	Binary
0.8	0.044	10	35	10	7	23	Binary
0.8	0.05	10	35	10	6	20	Binary
0.8	0.06	10	35	10	5	17	Binary
0.8	0.07	10	35	10	4	14	Binary
0.8	0.091	10	35	10	3	11	Binary
0.8	0.031	10	50	10	10	32	Gray
0.8	0.034	10	50	10	9	29	Gray
0.8	0.038	10	35	10	8	26	Gray
0.8	0.044	10	35	10	7	23	Gray
0.8	0.05	10	35	10	6	20	Gray
0.8	0.06	10	35	10	5	17	Gray
0.8	0.07	10	35	10	4	14	Gray
0.8	0.091	10	35	10	3	11	Gray
MULTI-OBJECTIVE PROBLEM							
Crossover	Mutation	Population	Generations	Runs	Bits	ℓ	Coding
0.8	0.031	40	100	1	10	32	Binary
CONSTRAINED MULTI-OBJECTIVE PROBLEM							
Crossover	Mutation	Population	Generations	Runs	Bits	ℓ	Coding
0.8	0.031	40	100	1	6	20	Gray

For each bit string of length ℓ, 10 runs of the algortihm were performed using the binary-coding and 10 runs using the Gray-coding. Best, median and the worst objective function values for each ℓ; the number of simulations needed to obtain these values and total number of simulations done by the algorithm are presented in table 2.

Table 2

Results for single-objective problem

BINARY CODING									
ℓ	dIC (€)			Simulations			Total Simulations		
	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
32	-2102	-2092	-1939	188	275	217	327	327	287
29	-2103	-2100	-1940	257	99	289	315	337	293
26	-2102	-2079	-2023	167	142	190	229	223	227
23	-2098	-2088	-1924	212	210	177	216	216	197
20	-2101	-2098	-2058	142	148	80	197	206	146
17	-2101	-2088	-1932	179	63	165	282	199	251
14	-2088	-2088	-1786	72	102	60	131	149	120
11	-2098	-2098	-1959	101	41	43	109	130	130
GRAY CODING									
ℓ	dIC (€)			Simulations			Total Simulations		
	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
32	-2102	-2072	-1939	368	302	250	377	310	315
29	-2102	-2091	-1928	279	231	206	290	289	293
26	-2102	-2087	-2001	150	253	165	219	257	220
23	-2101	-2086	-1932	206	238	162	262	261	255
20	-2101	-2098	-2015	102	186	40	202	234	157
17	-2096	-2096	-1915	105	157	119	180	206	178
14	-2088	-1994	-1885	130	83	75	132	147	146
11	-2098	-2098	-1824	35	56	35	98	98	106

In each run presented, the simulations needed to obtain the best solution was less than total simulations. These differences are presented in fig. 2.

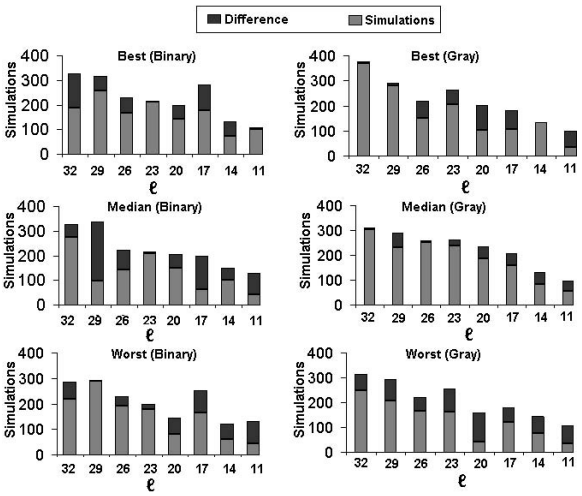

Figure 2 The differencences between simulations needed to obtain the best solution and total simulations done

In figures 3 and 4 the results of 10 runs is presented, where 10 bit-coding for continuous variables was used. For each run the best result obtained after 100, 200 and 300 simulations is shown.

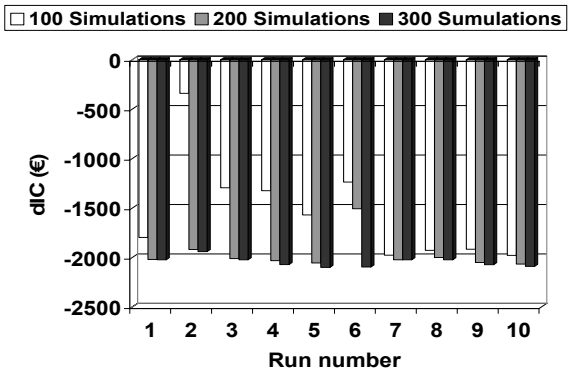

Figure 3 Results obtained after 100, 200 and 300 simulations using binary-coding

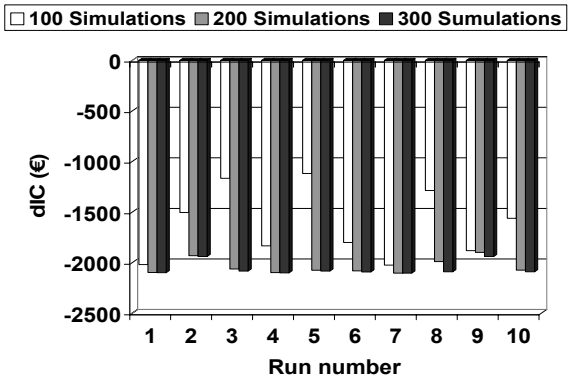

Figure 4 Results obtained after 100, 200 and 300 simulations using Gray-coding

When the accuracy of the decision parameters is decreased, a huge reductions in simulations needed is obtained, but the quality of the solution obtained does not change dramatically. Table 3 demonstrates this with results obtained using normal binary-coding. First row shows number of bits used to code continuous variables. Second row shows average of minimum values obtained after 100 simulations in 10 runs for each bit-string length. Second row shows average of minimum values obtained after runing algorithm 10 times for 50 or 35 generations (see table 1) for each bit-string length. Third row is the average number of simulations needed to reach the best solution in 10 runs.

Table 3 Average of minimum values in 10 runs using binary-coding

Bits	10	9	8	7	6	5	4	3
dIC (€)	-1619	-1611	-1622	-1794	-1831	-1771	-1998	-2056
dIC (€)	-2063	-2077	-2080	-2057	-2091	-2049	-1998	-2056
Simulations	260	220	177	194	192	191	108	134

USING GA AS INITIAL POINT GENERATOR FOR LOCAL SEARCH ALGORITHMS

Next the results using the hybrid algorithm are presented. Algorithm does global search using GA to create initial point for the Hooke-Jeeves (H-J) algorithm originally implemented in GenOpt [5]. Table 4 presents the parameters and the results obtained in four runs of the algorithm.

Table 4 Parameters and results for four runs of the hybrid algorithm implemented

		RUN1	RUN2	RUN3	RUN4
GA	Crossover	0.8	0.8	0.8	0.8
	Mutation	0.031	0.031	0.031	0.031
	Stopping criterion (simulations)	100	100	100	100
	Bits	10	10	10	10
	ℓ	32	32	32	32
	Coding	Binary	Binary	Binary	Binary
	GA Result (dIC €)	-1290	-1560	-1770	-2010
HOOKE-JEEVES	mesh size divider	2	2	2	2
	initial mesh size exponent	0	0	0	0
	mesh size exponent increment	1	1	1	1
	number of step reductions	8	8	8	8
	step	0.05	0.05	0.05	0.05
	Initial Point (dIC €)	-1290	-1560	-1770	-2010
	Stopping criterion (dIC €)	NO	NO	NO	NO
	Simulations	117	132	125	109
	Total simulations (GA+H-J)	217	232	225	209
	Final result (dIC €)	-2102	-2102	-2103	-2102

Results in table 5 state that by using GA as initial point generator for Hooke-Jeeves algorithm, the same or better result are obtained compared to the result in [1] with lower number of simulations.

OPTIMIZATION RESULTS FOR A MULTI-OBJECTIVE PROBLEM

Next it will be demonstrated how the implemented GA can be used to solve problems with more than one cost functions. Our first objective is to minimize the difference in investment dIC (€). Second objective is to minimize the annual space heating energy of the heating system.

Algorithms parameters are presented in table 1. Figure 5 shows all points visited by the algorithm. Variant of non-dominated sort is run on this set and set of results is obtained to this optimization problem. Results of the standard Omni-optimizer and the implemented GA are presented in figure 6. Using the external archive proposed, 1200% more solutions were obtained than standard Omni-optimizer procedure.

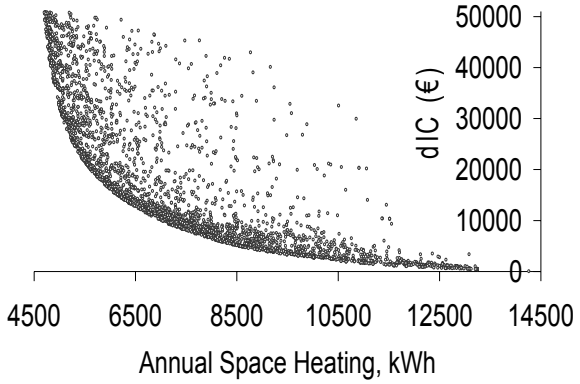

Figure 5 Points visited by algorithm

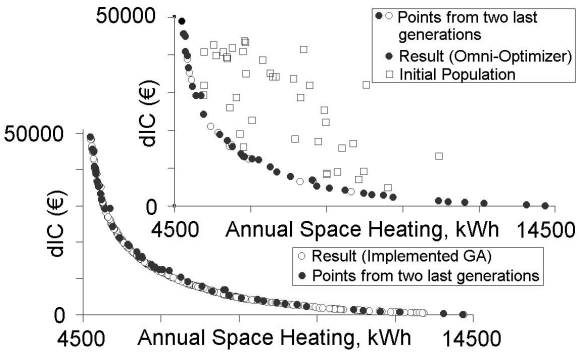

Figure 6 Results obtained for multi objective optimization problem.

Result of 502 non dominated points were obtained by the algorithm with 3200 simulations. Bookkeeping of evaluated solutions saved 800 simulations. Implemented GA was capable of obtaining diverse set of non-dominated solutions in single optimization run. Also extreme solutions with respect to both objectives were found by the algorithm.

OPTIMIZATION RESULTS ON CONSTRAINED MULTI-OBJECTIVE PROBLEM

A brute-force search method was implemented in [3] to check the results obtained by the optimization. The brute-force search is an exhaustive search that systematically enumerates all possible candidate solutions. In order to make the brute-force search feasible, the size of the problem was limited by using some indications from the optimization results. Results obtained by the current study are compared with the above mentioned brute-force results.

The maximum value for dIC in the feasible brute-force was less than 6000 Euro. Therefore, maximum difference in investment cost (dICmax= 6000 Euro) is considered as a constraint on the objective function.

Implemented algorithm has constraint handling strategy that needs no parameters from designer. We will run algorithm with parameters presented in table 1. Results are presented in in figure 7.

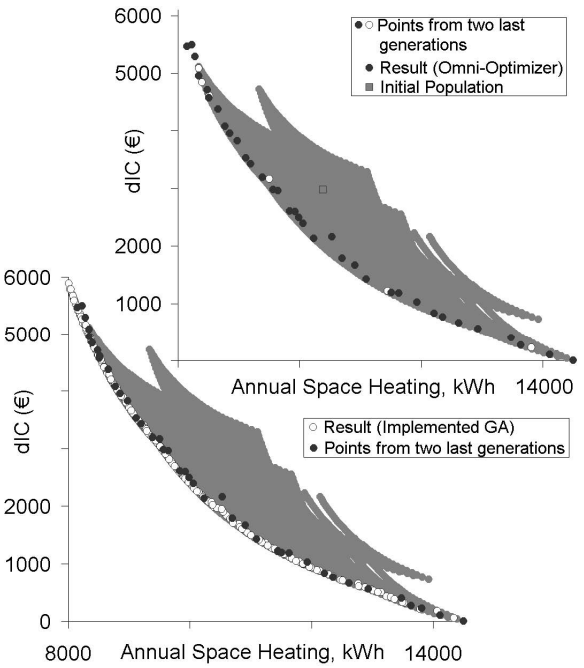

Figure 7 Solutions to constrained multi-objective problem

As can be seen from the figure 7 the implemented algorithm is capable of obtaining a diverse set of feasible non-dominated solutions in single optimization run. These solutions are in True Pareto-front or very close to it. Also extreme solutions with respect to both objectives were found by the algorithm. Number of non dominated solutions obtained was 944 in 2193 simulations. Bookkeeping of evaluated solutions saved 1807 simulations.

CONCLUSION

As a result of this study a collection of 26 evolutionary strategies is implemented with three different coding schemes. This collection is planned to be added to GenOpt's [5] algorithm library.

Defining a stopping criterion for GA is shown to be very important if number of simulations needed is to be decreased. Actual number of simulations needed to reach optimal or near optimal solution is in general much less than simulations done with naive stopping criterion used in this study.

Implemented GA was able to obtain equal results in several runs compared to result in [3]. However, because of the stochastical behavior of the GA it is also concluded that there are no guarantees that GA will reach the optimal solution in every run. However it is shown that all results obtained with GA are close to optimal in single-objective problem. Simulations done by GA are in general more than in [3] where a hybrid algorithm was used.

Hybrid algorithm was implemented to decrease simulations needed in a single-objective problem. Near optimal solution, convergence of GA slows down dramatically, thus use of the hybrid algorithm is shown to give better results in context of a quality of the result and the simulations needed to reach the result. In every run of the hybrid algorithm better or equal result were obtained than in [3]. Results were better or equal in objective function value and the number of simulations needed to obtain the solution was much less than in [3].

In multi-objective problem it is shown that the algorithm is capable to generate a diverse set of non-dominated solutions and is able to capture extreme solutions to all objective functions. In the constrained multi-objective problem it is shown that most of the

obtained solutions are in true Pareto-front or very close to it.

Use of external archive is shown to improve the diversity, convergence and number of solutions obtained by both NSGA-II and Omni-optimizer.

ACKNOWLEDGEMENT

The authors would like to acknowledge the financial support of the Finnish National Technology Agency (TEKES), as part of the MASI programme, as well as the following supporting companies: Optiplan Oy, Pöyry Building Services Oy, Saint-Gobain Isover Oy, Skanska Oy, and YIT Oyj.

REFERENCES

[1] Deb K., Pratap A., Agawal S., Meyarivan T. A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. Kampur Genetic Algorithms Laboratory (KanGAL). Indian Institute of Technology Kampur.

[2] Deb K., Tiwari S. Omni-optimizer: A Procedure for single and Multi-Objective Optimization. Kampur Genetic Algorithms Laboratory (KanGAL). Indian Institute of Technology Kampur.

[3] Hasan, A., Vuolle, M., Siren, K. 2008. Minimisation of life cycle cost of a detached house using combined simulation and optimisation. Building and Environment, Volume 43, No 12, December 2008, Pages 2022-2034.

[4] IDA ICE (IDA Indoor Climate and Energy) http://www.equa.se/eng.ice.html

[5] Wetter M. GenOpt® Generic Optimization program. User Manual Version 2.0. Technical Report LBNL-54199. Building Technologies Program, Simulation Research Group, Lawrence Berkeley National Laboratory (http://gundog.lbl.gov/GO/).

[6] Deb K. 2001. Multi-Objective Optimization using Evolutionary Algorithms. Wiley.

[7] Michael Wetter, Jonathan Wright
A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization
Building and Environment
Volume 39, Issue 8, August 2004, Pages 989-99

APPENDIX A FLOWCHART OF IMPLEMENTED GA