Using Kriging regression to improve the stability and diversity in NSGA-II

Michael Wood and Matthew Eames
College of Engineering, Mathematics and Physical Sciences, University of Exeter, UK

Abstract

Non-dominated sorting genetic algorithm version 2 (NSGA-II) is a multi-objective optimisation method. NSGA-II is often used to optimise the design of building. This paper details small improvements to this algorithm using 'fitness approximation' methods.

Fitness approximation is used speed up the conversion of NSGA-II. Radial basis functions networks have been shown to be useful for this. Although there are many types of fitness approximation function that could be use for this purpose, Kriging methods have not yet been tested.

In this paper, Kriging models are compared to standard NSGA-II. The results show that Kriging-based fitness approximation slightly improves upon standard NSGA-II. More work is needed to test this method on different building types as well as more complex problems, such as those associated with HVAC design.

Introduction

Optimising buildings using computational techniques is becoming more common (Nguyen, Reiter, & Rigo, 2014). Optimisation methods are particularly common in building facade design research (Zemella, De March, Borrotti, & Poli, 2011) and residential building envelope design (Tuhus-Dubrow & Krarti, 2010).

Genetic algorithms (GAs) are popular optimisation tools, and are used widely in engineering (Ooka & Komamura, 2009; Wright & Alajmi, 2005). Of the GA algorithms in use, NSGA-II has been shown to be a very effective tool and is used in a wide variety of building optimisation software including TYNSYS (The University of Wisconsin, 2016), IES (Integrated Environmental Solutions Ltd., 2017) and Design Builder (Design Builder Software Ltd., 2017) as well as other applications (Hamdy, Hasan, & Siren, 2013).

Other optimisation methods that have proven useful in building design include artificial neural networks (Magnier & Haghighat, 2010).

In this paper, we examine whether NSGA-II can be improved using *universal Kriging* as a fitness approximation function.

The aim of this work is to test if Kriging-based surrogate modelling can be used to improve the results of NSGA-II. We test this on a simple building design problem.

Background

About NSGA-II

NSGA-II is a multi-objective optimisation algorithm based on genetic algorithms. It is commonly used to solve multi-output optimisation problems.

NSGA-II has also found use in building design software. Figure 1shows an example taken from Design Builder's promotional literature. It show the output of an optimisation minimising cost and CO_2 emissions.

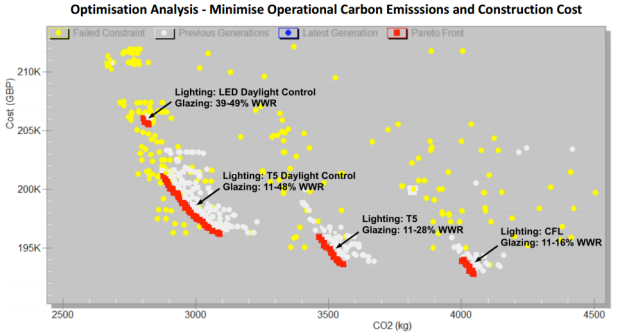

Figure 1: Example solutions to balance cost and CO2 emissions (adapted from Cocking & Tindale, 2014)

The algorithm starts with a random slection of potential solutions referred to as *population*. It then uses this population and its results to generate a new population. If the algorithm is working well, each new population will improve on the previous population.

NSGA-II also has a property called *elitism*. This means that it always keeps the best results between each population iteration. The algorithm achieves this by combining the parent and offspring populations and selecting the best from the *combined* population.

Each generation of a new population is called a *generation*. As the algorithm progresses through each subsequent generation, it should converge on an ideal set of results - the *Pareto* front.

Generating new populations

NSGA-II creates better solutions by mimicking evolutionary processes. It starts this process by mapping the numerical inputs to the model (U-values, air permeability *etc.*) onto a virtual 'chromosome'.

These 'chromosomes' are a binary string representation of each variable. For example a variable 133 might be

represented as 10000101. Representing the variables in this way allows the values to be mutated much like the way genes are in DNA. However, instead of manipulating the genetic code, NSGA-II manipulates the binary strings that represent the model's input. It does this using three stages:

1. Tournament selection
2. Mutation
3. Cross-over

Tournament selection

The purpose of tournament selection is to create a 'mating pool' from which the offspring population Q will be generated. This is achieved by first picking a parent population P at random. Once the parent population is created, NSGA-II runs the following steps. It;

1. randomly selects n sample
2. s from the parent population (where n is the tournament size);
3. adds the n samples to the mating pool; and
4. repeats steps 1 and 2 until size of mating pool is equal in size to P (the size of the parent population).

The tournament selection phase produces the first *offspring* population. These offspring candidates are then altered through crossover and mutation.

Crossover

The crossover phase allows the swapping of genetic information between members of the mating pool. 'Parent' chromosomes from the mating pool are selected at random for crossover as are the specific genes on the chromosome of each parent. The left hand image in Figure 2 shows the crossover of genetic information on the third gene of two parents from the mating pool:

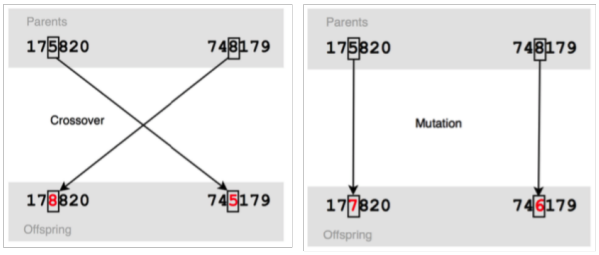

Figure 2: Crossover and mutation operations

Mutation

Mutation is an operation where one (or more) of the elements of a gene are changed at random. The location of the change and its magnitude are chosen at random. The right hand image in figure 2 illustrates a mutation of magnitude 2 on the third gene to two parents in the mating pool.

NSGA-II overview

NSGA-II uses the steps described above to generate each subsequent population. An overview of this process is shown below:

1. Create an initial population of buildings (P): These are chosen at random. The population of buildings is simulated to obtain the relevant results for each building.
2. Use the population P to generate a new set of buildings Q using tournament selection, crossover, and mutation.
3. Combine the results of P and Q into a new larger population R. Evaluate the population R with the simulator.
4. Find the best solutions in R using non-dominated sorting and crowding distance sorting. This creates the new population at time $t+1$, P_{t+1}.

A graphical representation of this process is shown in figure below:

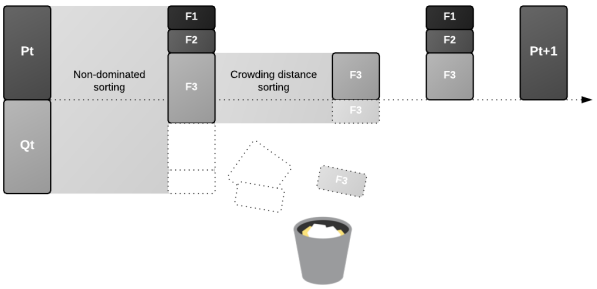

Figure 3: Summary of the NSGA-II process (Deb et al., 2002)

How can regression methods help improve NSGA-II

The speed at which the algorithm improves is based on the qualities of the new population Q. The better the population Q is, the quicker the algorithm will converge on the true Pareto front.

One way of improving Q is to make the population size of Q much bigger (Brownlee & Wright, 2015). In the standard mode of operation, the population size of Q is the same size as that of the initial population P. If Q is made k times bigger, then the rate at which the population improves with each iteration will quicken.

An increase in the size of Q causes problems. Increasing the size of Q by an order of k means that the number of simulations required at each iteration is also increased by k. Clearly, this is not worth it, since a k-fold increase in simulations in Q is unlikely to provide superior results to a k-fold increase in the number of population *generations*.

Regression is a way of solving this problem - instead of running the extra Q candidates through the simulator, we can instead evaluate them with a surrogate model. This allows us to generate large populations of Q' that are much larger than Q without much computational overhead. The population in Q' is sorted and ranked in the same way as Q. In the same way, only the top n

survive. These rejoin the population P as in the standard NSGA-II and the algorithm continues (Figure 3).

The surrogate model will continue to be a better and better predictor of the output of Q', since the more we iterate through the NSGA-II process, the more information we have on the simulator's output. Because of this, the model will get better as the number of evaluated populations increases.

Kriging regression methods

Kriging is a sophisticated linear regression method. It represents the output of a simulator function $f(\boldsymbol{x})$ as a multivariate Gaussian process $\hat{f}(\boldsymbol{x})$:

$$\hat{f}(\boldsymbol{x}) \sim \mathrm{GP}(m(\boldsymbol{x}), v(\boldsymbol{x}, \boldsymbol{x}')) \qquad (1)$$

Although that output of the Gaussian process is essentially random Gaussian noise (with a mean of $m(\boldsymbol{x})$ and a variance of $v(\boldsymbol{x}, \boldsymbol{x}')$), this doesn't mean that we think the original simulator output is random. Instead, we are using the Gaussian process to allow us to express uncertainty in the output:

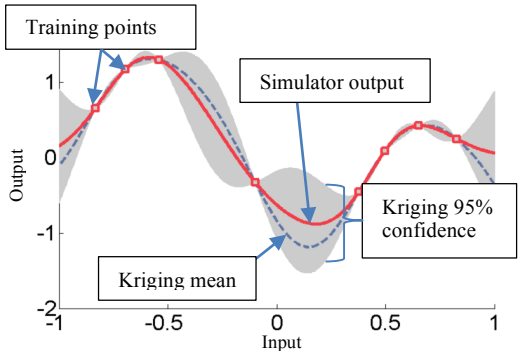

Figure 4: Kriging example

The mean and variance functions of the Kriging model are estimated using a training set (◘ shown in Figure 4). Standard functions govern the mean and variance. However, these functions require the tuning and estimation of hyperparameters, whos derivation would be too lengthy to detail here. There interested reader is referred to *Gaussian Processes for Machine Learning* (Rasmussen & Williams, 2006).

Kriging has several advantages over standard linear regression methods:

1. It can make predictions of its own uncertainty
2. The convergence is faster[1]
3. The regression prediction $\hat{f}(x)$ is equal to the 'true' output $f(x)$ for points in the training set $\mathbf{D}$[2]
4. It can capture non-linear processes with a high order of complexity with relatively few training simulations

[1] Convergence in this case refers to convergence on the objective function, not the optimum solution.

[2] This is not true of other linear regression methods, like polynomial regression, where the closeness of fit is limited by the order of the regression equation

We use Kriging to evaluate the large populations in Q', which are then reduced in size (based on the best solutions) to provide a better offspring populations in Q. Our aim is to test whether the unique properties in the Kriging model (as opposed to other regression methods) leads to a significant improvement in the performance of NSGA-II.

Method

In adding the fitness approximation to the algorithm, we add an additional step to the NSGA-II process. The step is added after the parent population but before the offspring population is created. This is where we create the intermediate offspring population Q'.

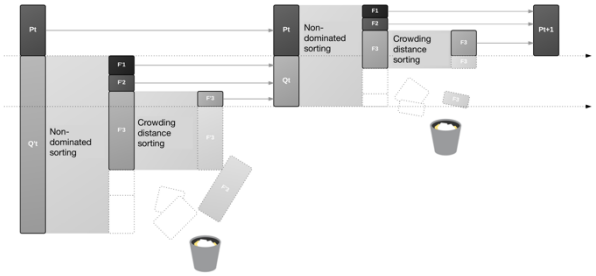

Figure 5: Schematic showing the Kriging-assisted NSGA-II approach

In this additional step we dont evaluate the fitness of the population with the simulator, we evaluate it with an emulator instead. This allows us to evaluate much larger candidate populations and our hypothesis is that this additional layer of population screening will lead to better convergence on the Pareto front. We refer to two different approaches as *standard NSGA-II* and *Kriging-assisted NSGA-II*.

For the Kriging-assisted NSGA-II, the initial population is selected using a Latin Hypercube sampling method (McLeod, Hopfe, & Kwan, 2013; Stein, 1987). We use this method because it makes it more likely that the initial emulator (on which the whole analysis is built) is more accurate (Loeppky, Sacks, & Welch, 2009).

The building model

The building model use for a test analysis is a simple office building (Figure 6).

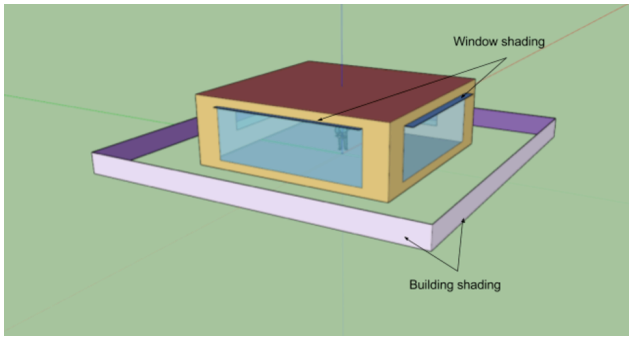

Figure 6: Office building used in the model

The building's construction materials are detailed in table 1.

Table 1: Material properties of the building

Name	Thickness (m)	Conductivity ($W.m^{-1}K^{-1}$)	Density (kgm^{-3})	Specific heat ($Jkg^{-1}K^{-1}$)
Membrane	0.0001	1.000	1100	1000
Plasterboard	0.0125	0.210	700	1000
Floor insulation	0.1100	0.025	700	1000
Concrete floor	0.1000	2.300	2300	1000
Chipboard	0.0200	0.130	500	1600
Carpet	0.0100	0.040	160	1360
Tiles	0.0127	0.840	1900	800
Concrete	0.2350	1.400	2100	840
Wall insulation	0.0840	0.030	43	1210
Roof insulation	0.0670	0.030	43	1210
100 mm brick	0.1000	0.890	1920	790
Brick partitions	0.0040	0.890	1920	790

The lights were operational between the hours of 7am and 7pm. They were set to provide a minimum of 500 lux on the working plane (centre of the room) and had a density such that the total power was 5 Wm^{-2}.

The occupancy schedule model used in the model is as shown in Figure 7.

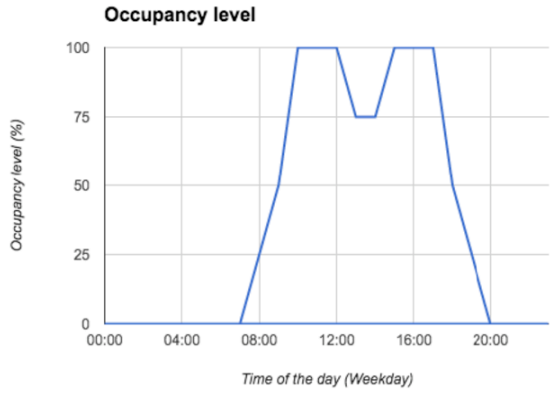

Figure 7: Occupancy schedule used in the model

The office is assumed to have 5 occupants. The materials are used in the construction as follows:

- Wall construction: 100mm brick / insulation / concrete
- Roof construction: Tiles / Membrane / insulation / plasterboard
- Floor construction: InsulationFloor / ConcreteFloor / Cavity / Chipboard / Carpet
- Windows construction: Low emissivity double glazing (6 mm/13 mm/6 mm) Argon filled (with 25% equivalent openable area for natural ventilation)

The ranges of the variables used in the building model are given in the appendix. The other setup parameters are:

- Heating set point: *21°C*
- Activity level per person: *80 Watts*
- Calculation detail - time step: *12 steps per hour*

The limits of the variables that can be chosen by each algorithm are shown in Table 2. We assume a uniform distribution of potential solutions for each variable (*i.e.* no prior weighting).

Table 2: Ranges of the variables used in the computer model

Real Name	Min	Max
Aspect ratio	0.50	1.0
U-value walls (Wm^{-2})	0.10	0.5
U-value roof (Wm^{-2})	0.10	0.5
Glazing north (%)	0.00	0.9
Glazing south (%)	0.00	0.9
Glazing east (%)	0.00	0.9
Glazing west (%)	0.00	0.9
Window U-value (Wm^{-2})	1.00	3.0
Concrete thickness (m)	0.05	0.5
Overhang north (m)	0.10	0.5
Overhang south (m)	0.10	0.5
Overhang east (m)	0.10	0.5
Overhang west (m)	0.10	0.5

Weather data

We used the new CIBSE Design Summer Year for London DSY1 (Eames, 2016). Note that any other weather file could have been used.

Optimisation criteria

We use the NSGA-II algorithms to generate a range of solutions that minimise both the lighting energy use and the overheating.

Overheating and lighting are conflicting design criteria. They compete with each other because and increase in glazing will mean that less lighting energy is required. However, a greater proportion of glazing has been linked to increased overheating. Conversely, less glazing mean less overheating, but it also means that more lighting energy will be required. The aim of the optimisation process is to generate a range of options that strike a balance between these two requirements.

We measured overheating by counting the number of hours where $\Delta T > 0$ (based on *Criterion a)*, which is taken from CIBSE TM 52 (Nicol, 2013)). (ΔT is greater than 0 when the internal temperature is above the comfort temperature.) We measured the lighting by measuring the total annual lighting energy use (kWh).

Genetic algorithm setup

The NSGA algorithms were used with the following settings:

Standard NSGA-II

- Population size: 20
- Crossover probability: 0.5
- Mutation probability: 0.4
- Tournament size: 3

Kriging Enhanced NSGA-II

- Population size: 20
- Crossover probability: 0.7
- Mutation probability: 0.4
- Tournament size: 3

Each analysis had a budget of 300 simulations and was repeated nine times. This allowed us to check for consistency in the results as well as monitoring the progress of each of the analysis runs.

Note that the crossover probabilities are different between the Standard NSGA-II and the Kriging enhanced version. We found that increasing the crossover probability by a small amount significantly improved the output of the Kriging emulator. We need to complete a more extensive analysis of each of the options. This is again the subject of further work.

Measuring the performance: Hypervolumes and spread

As well as tracking each of the individual runs through each generation, we also monitored the *hypervolume* and *diversity* in the results.

The hypervolume and diversity are the two key measures of the quality of the end result. The hypervolume can be thought of as a measure of how close the solution is to the 'true' Pareto front and the diversity is a measure of how 'evenly spread out' the solutions are.

An illustration of two different levels of diversity are in Figure 8. An illustration of the hypervolume is shown in Figure 9.

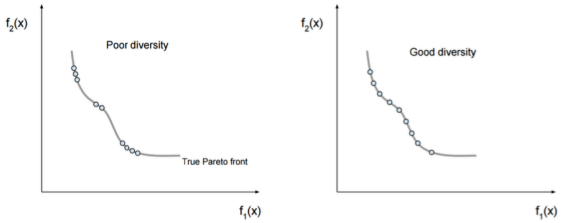

Figure 8: Examples of diversity in the result's outputs

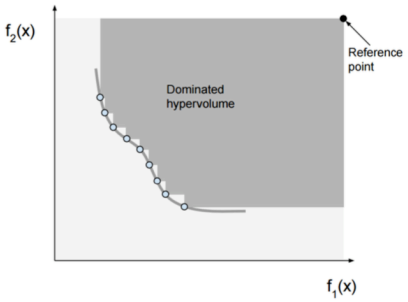

Figure 9: Hypervolume definition

A good result will have a higher hypervolume and a higher spread coefficient (the options are as evenly distributed as possible across the Pareto front).

The size of the hypervolume relies on the position of the reference point. For our purposes, the reference point is located at $x = 20$ and $y = 1300$.

The spread is a single value measure which is dependent on the 'spread area' in the output space as shown in Figure 10.

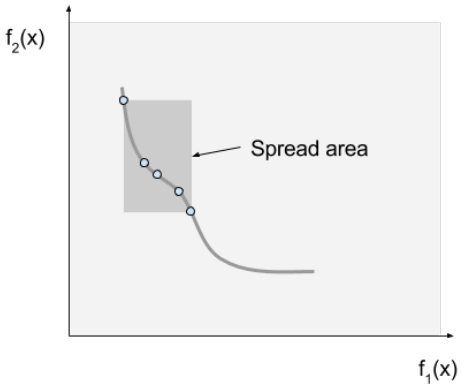

Figure 10: Spread definition

Results

Figure 11 shows the output of one of the analyses after 15 generations for both standard NSGA-II and Kriging-assisted NSGA-II.

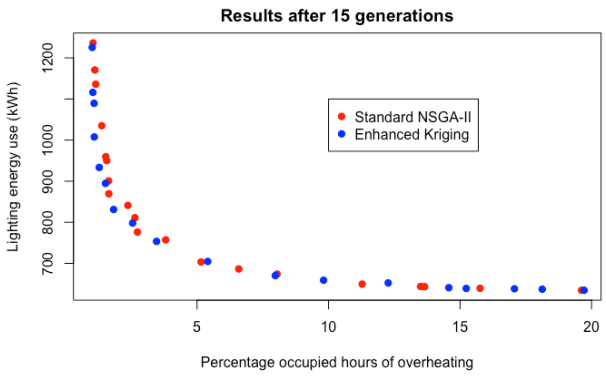

Figure 11: Results after 15 generations (standard and Kriging enhanced NSGA-II)

The results of the Kriging-enhanced NSGA-II are slightly better in both spread and in the closeness to the

Pareto front. This is true for all nine repetitions of the algorithm. The hypervolume and spread progress is shown in Figure 12 to Figure 15.

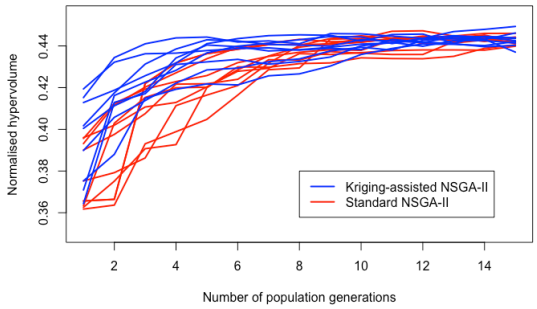

Figure 12: Hypervolume progress

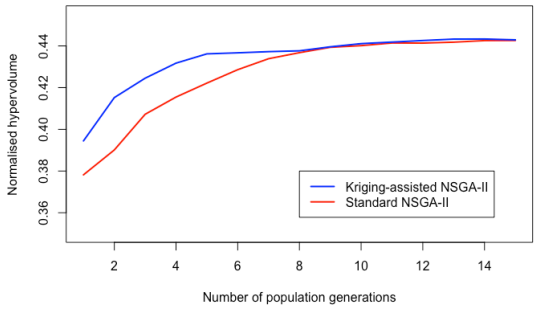

Figure 13: Mean hypervolume progress (over 9 repetitions)

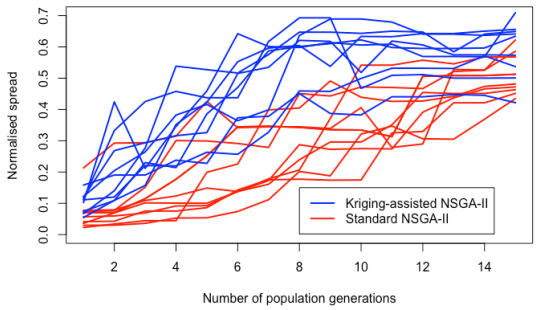

Figure 14: Spread progress

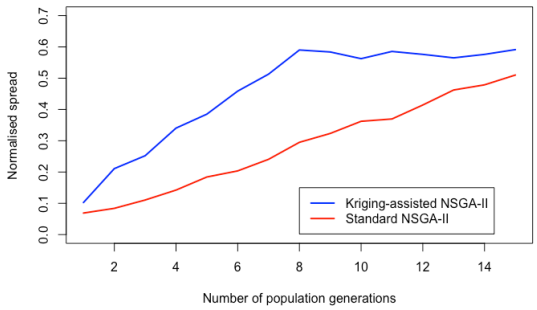

Figure 15: Mean spread progress of Kriging assisted NSGA-II vs. standard NSGA-II.

Discussion and conclusion

The results show that the Kriging enhanced NSGA-II performs well in comparison to the standard NSGA-II. Both the mean hypervolumes and the mean spreads show differences (though the different in the spread is more significant).

For the hypervolume, the Kriging-assisted method is superior up to around eight generations. The standard NSGA-II then catches up.

After 15 iterations, the Kriging assisted NSGA-II has a much better spread of results. It also appears to get a much better spread after fewer iterations. Further work will be required to determine whether this is case for other building design problems.

When considering the improvements offered by the Kriging-assisted design, we need to consider the additional computational costs.

The Kriging model adds two additional computational loads. There are;

a) interrogating the Kriging emulator to evaluate each; and

b) creating the Kriging model at each generation.

Interrogating the emulator its outputs extremely quick. For a typical desktop computer, it takes around 10^{-6} seconds. The building simulator used in the assessment takes approximately 30 seconds to run, so in this case the emulation time is insignificant.

We also need to consider how long it takes to create the Kriging model. The time it takes to create the Kriging model is (roughly) proportional to n^2 (where n is the number of training samples). The Kriging model is built using all of the samples that have passed through the real building simulator. So, as the the number of generations increases, the size of the Kriging training set also increases. There is therefore a problem for using this model with either a large number of population generations, large populations or both.

The Kriging assisted algorithms take longer to run. The total simulation time is a combination of the time taken to;

1. run the simulations;
2. perform the genetic operations in the NSGA model; and
3. rebuild the Kriging emulators between generations

The standard NSGA-II method only runs steps one and two, the Kriging assisted method needs all three.

The time taken to run the simulations is around 30 seconds. This time is much greater than the time taken to execute steps two and three (typically << 1 second). The extra time added by the Kriging processes is insignificant. For other simulation optimisation problems, where the simulation time is much lower, the

extra computation time will begin to become more significant.

Further work should include;

- examining problems where there are more than two outputs objectives;
- testing the efficacy of the different algorithms on different building problems (*e.g.* examining ventilation and HVAC strategies where the optimisation geometeries can be extremely complex);
- running the algorithms for different building and climate types.

Further work should also consider investigating the 'settings' for crossover, mutation and tournament selection used in both algorithms. There are many possible combination and permutation of these settings, which all need to be tested statistically on a range of problems.

The method should also be compared to 'pure' Kriging-based multi-objective optimisation. Methods such as ParEGO (Knowles, 2006) should also be investigated.

Not withstanding the need for further work, the results show that Kriging-based fitness approximation shows promise for improving NSGA-II. Further work is needed to strengthen this finding.

Acknowledgements

The authors would like to thank for EPSRC for funding this research [Ref: EP/M021890/1]

References

Design Builder Software Ltd. (2017). Design Builder. Retrieved from http://www.designbuilder.co.uk/

Eames, M. (2016). An update of the UKs design summer years: Probabilistic design summer years for enhanced overheating risk analysis in building design. *Building Services Engineering Research and Technology*, *37*(5), 503–522. https://doi.org/10.1177/0143624416631131

Hamdy, M., Hasan, A., & Siren, K. (2013). A multi-stage optimization method for cost-optimal and nearly-zero-energy building solutions in line with the EPBD-recast 2010. *Energy and Buildings*, *56*, 189–203. https://doi.org/10.1016/j.enbuild.2012.08.023

Integrated Environmental Solutions Ltd. (2017). IES (www.iesve.com). Retrieved from http://www.iesve.com/

Knowles, J. (2006). ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Trans. Evol. Comput.*, *10*(1), 50–66.

Loeppky, J. L., Sacks, J., & Welch, W. J. (2009). Choosing the Sample Size of a Computer Experiment: A Practical Guide. *Technometrics*, *51*(4), 366–376. https://doi.org/10.1198/TECH.2009.08040

Magnier, L., & Haghighat, F. (2010). Multiobjective optimization of building design using TRNSYS simulations, genetic algorithm, and Artificial Neural Network. *Build. Environ.*, *45*(3), 739–746.

McLeod, R. S., Hopfe, C. J., & Kwan, A. (2013). An investigation into future performance and overheating risks in Passivhaus dwellings. *Building and Environment*, *70*, 189–209. https://doi.org/10.1016/j.buildenv.2013.08.024

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. https://doi.org/10.1016/j.apenergy.2013.08.061

Nicol, F. (2013). TM52: The limits of thermal comfort: avoiding overheating in European Buildings.

Ooka, R., & Komamura, K. (2009). Optimal design method for building energy systems using genetic algorithms. *Building and Environment*, *44*(7), 1538–1544. https://doi.org/10.1016/j.buildenv.2008.07.006

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning. International journal of neural systems* (Vol. 14).

Stein, M. (1987). Large Sample Properties of Simulations Using Latin Hypercube Sampling. *Technometrics*, *29*(2), 143–151. https://doi.org/10.1080/00401706.1987.10488205

The University of Wisconsin. (2016). A TRaNsient SYstems Simulation Program (TRNSYS).

Tuhus-Dubrow, D., & Krarti, M. (2010). Genetic-algorithm based approach to optimize building envelope design for residential buildings. *Building and Environment*, *45*(7), 1574–1581. https://doi.org/10.1016/j.buildenv.2010.01.005

Wright, J., & Alajmi, A. (2005). The robustness of genetic algorithms in solving unconstrained building optimization problems. *Proceedings of Building Simulation*, *5*, 1361–1368.

Zemella, G., De March, D., Borrotti, M., & Poli, I. (2011). Optimised design of energy efficient building façades via Evolutionary Neural Networks. *Energy and Buildings*, *43*(12), 3297–3302. https://doi.org/10.1016/j.enbuild.2011.10.006