INTERNATIONAL BUILDING PERFORMANCE SIMULATION ASSOCIATION ENGLAND

Proceedings of BSO-V 2020
5th Building Simulation and Optimization Virtual Conference
21-22 September 2020
Loughborough, UK

Decision Support in Algorithm Selection for Generic Optimisation

Vishak Dudhee[1], Fathi Abugchem[1], Vladimir Vukovic[1]
[1]Teesside University, Middlesbrough, United Kingdom

Abstract

This paper presents the development of an algorithm-selection framework supported by a new intuitive user interface for the generic optimisation tool, GenOpt. The framework consists of an algorithm-selection flowchart to help identify relevant algorithms depending on the nature of the problem, followed by an algorithm-selection matrix which evaluates the algorithms' suitability based on the user requirements. The algorithm selection framework acts as a decision support system to allow the user to select the most appropriate and effective optimisation algorithm for a given problem. Such a procedure improves decision-making, limits the algorithm selection errors and helps the user to achieve solutions closer to the Pareto optimum. The selection framework is supported by a user interface, developed in C++ and compatible with GenOpt, that allows users who do not have prior coding knowledge to use GenOpt successfully. The developed interface presents the user with the most relevant optimisation algorithms from those available in the programme. It allows the user to easily modify algorithmic variables in a user-friendly environment. The novelty of the approach is reflected in the built-in knowledge and intelligence in the pre-selection of optimisation algorithms, which are tailored to specific user-defined problems. This, consequently, improves the overall optimisation results by allowing the user to better understand the optimisation algorithm and its variables.

Introduction

Careful consideration of multiple design parameters is required in order to design energy-efficient systems. Dynamic whole-building simulation programs can be used to model the impact of certain parameters on system performance, such as energy consumption and thermal losses (Crawley et al. 2000). However, the interactions between several design parameters can make optimisation a complex task, with many possible parameters whose relations with system performance may be difficult to understand (Wortmann and Nannicini, 2017). Optimisation algorithms can identify optimal parameters for a defined cost function, such as energy performance.

Generic optimisation programmes, such as GenOpt, allow for automatic, multidimensional optimisation of system simulation models, which eventually leads to efficient system designs (Wetter, 2000). Such programmes allow customisation of optimisation algorithms and therefore can be used as an optimisation algorithm development environment. Algorithm selection and tuning have a huge impact on the optimisation performance both on the optimality of the solutions and the speed of convergence.

Problem Statement

Generic Optimisation program GenOpt's documentation includes guidance on selecting the optimisation algorithms and setting their parameters, but such guidance is not integrated into the optimisation programme graphical user interface. Therefore, the user needs to pick and choose the optimisation algorithm and determine their relevance to the simulation model based on prior knowledge or external literature. Once the user selects the relevant algorithm, the algorithm set-up processes are not direct but involve finding the appropriate programme file(s) and changing the relevant codes. Most practitioners neither have the expertise in optimisation algorithms needed to make an informed decision nor the programming knowledge to amend the command codes with the selected optimisation algorithm.

Optimisation Processes

Optimal selection of optimisation algorithms depends upon the type of optimisation problems.

Non-linear Optimisation

Non-linear optimisation problems can be classified into three main categories: one-dimensional unconstrained problems, multidimensional unconstrained problems, and multi-dimensional constrained problems (Antoniou and Lu, 2007). One-dimensional optimisation methods can be classified into search methods and approximation methods. In search methods, a lower and an upper interval boundary is established and then repeatedly reduced based on functional evaluations until a reduced boundary interval which is sufficiently small is obtained. The centre of such lower interval is then assumed to be the optimum. Search methods can be used for any function and differentiation is not necessary, unlike the approximation methods, where an approximation of the function, represented as a low-order – usually second- or third-order – polynomial is assumed. The objective function is then analysed with elementary calculus, leading to an approximate value of the function domain. The interval is reduced, and the process is repeated until an adequately precise value of the function domain is found. In one-dimensional optimisation methods, the function is required to be continuous and differentiable. Multi-dimensional optimisation methods are analogous to one-dimensional, but they sometimes can be inefficient. Multi-dimensional optimisation algorithms simultaneously consider multiple, potentially conflicting, objective functions (Wortmann and Nannicini, 2017). A problem with multiple objectives may not have a clear-cut solution. This is because the set of all non-dominated solutions, known as the Pareto front, may be infinite in size and difficult to accurately represent. It is impossible to improve an objective value for a non-dominant solution without losing in other objective values. Therefore, the application of multi-dimensional optimisation methods is limited to problems where gradient information is unavailable or difficult to obtain.

Heuristic Optimisation

Heuristic optimisation is a problem-solving method used to increase speed by sacrificing precision. (Gabbar, 2016). Metaheuristic optimisation is a higher-level heuristic with the purpose of identifying or generating an adequate solution to an optimisation problem when incomplete or flawed information is available or when there is limited computation capacity. They sample a group of solutions that are too large to be thoroughly sampled using conventional methods. Metaheuristics make assumptions about the optimisation problems, and so they may be useful for a variety of problems (Osman and Kelly, 1996; Dey, 2017). Metaheuristics, unlike optimisation algorithms and iterative methods, do not guarantee that a globally optimal solution can be generated in all problems. Many metaheuristics apply a form of stochastic optimisation, meaning that the solution relies on a set of randomly generated variables. By searching through a wide range of possible solutions, metaheuristics can often find reasonable solutions with minimal computational effort compared to optimisation algorithms, iterative methods, and simple heuristics.

Specific heuristic methods do not always perform effectively with alternative problem domains without considerable modification (Drake et al., 2020). The term “hyper-heuristic” can be defined as a high-level automated search methodology that explores a search space of low-level heuristics or heuristic components, to solve computationally difficult problems. Hyper-heuristics operate on a search space of heuristics rather than problem solutions themselves (Burke et al., 2013). This feature provides the potential for increasing the level of generality of search methodologies. These can be used to solve more complex real-world problems. Because the search strategy components of a hyper-heuristic only consider problem domain-independent information, hyper-heuristic methods can be easily applied in various problem domains given that the problem-specific algorithm components are accessible to the user.

Fitness Landscape

Fitness Landscape is a type of model that is used in both biology and social science to visualise the relationship between genotypes and reproductive success (Marks, Gerrits and Marx, 2019). Similarly, in optimisation, fitness landscape is used to describe and analyse the geometry of the search space from the point of view of local search algorithms, such as evolutionary algorithms (EAs) or single solution based local search (Leprêtre et al., 2019). The fitness landscape produces an image that represents the search space and helps design optimisation algorithms. It shows a representation of the problem structure using a set of metric features to measure and compare the search difficulty of various possible representations, local search operators, or objective functions. Fitness Landscape Analysis (FLA) helps the user to better understand the problem type that is being solved in practice rather than using mathematical test functions as often used in benchmarks (Waibel et al., 2019). Usage of such a technique for system energy optimisation can improve system design as it increases the emphasis on relevant design parameters.

Generic Optimisation Program

Overview

GenOpt is a generic optimisation program developed for system optimisation by Lawrence Berkeley National Laboratory. For a given system, GenOpt finds the user-selected values or parameters that can minimise the objective function, ultimately leading to the best operation of the system (Wetter, 2001). The objective function can be calculated by an external simulation program and integrated into GenOpt by modifying a configuration file of simulation program output comprised of a text-based input and output. To carry out the optimisation, GenOpt automatically generates the input files for the simulation program based on input template files specific to the program being used. GenOpt then starts the simulation program, checks for potential errors, reads the value of the minimised function and determines the input parameters for the next run. GenOpt repeats this process iteratively until a minimum is identified, displaying the results onto the graphical user interface during the optimisation process. GenOpt and the external simulation program exchange data solely through text files. GenOpt automatically creates the new input files for the simulation program based on input template files. To generate these template files, the user must copy the simulation input files and replace the numerical values of the independent variables, which will be modified with keywords. The keywords are then converted to the corresponding numerical values and the simulation input files are generated. GenOpt can write text input for any given simulation program using this method. The user may specify how to start the simulation program in a configuration file and indicate where in that file GenOpt can locate the cost function’s current value. This allows any external program to be coupled with GenOpt without either program needing to be modified or recompiled. The only requirement is that the external program must use text files to read its input and write the cost function value and any potential error messages.

GenOpt Optimisation Algorithms

The following optimisation algorithms are implemented in GenOpt:

- Generalised Pattern Search algorithms (GPS) with
  - Hooke-Jeeves (GPSHJ)
  - Coordinate Search algorithm (GPSCS)
- Particle Swarm Optimisation algorithms (PSO)
- A hybrid global optimisation algorithm that uses Particle Swarm Optimisation for global optimisation, and Hooke-Jeeves for the local optimisation (GPSPSOCCHJ).
- Discrete Armijo Gradient algorithm (DAG).
- Nelder and Mead's Simplex algorithm.
- Golden Section and Fibonacci.

The algorithms that can be used for parametric studies include:

- Mesh generator to evaluate a function on all points that belong to a mesh with equidistant or logarithmic spacing between the mesh points.
- Parametric search where only one independent variable is varied at a time.

Methodology

Algorithm Selection

The GenOpt user manual describes the best algorithm selection process based on the type of problem in the form of text (Wetter, 2016). To better understand the algorithm selection process, the different problems were classified by their type and their recommended algorithm tabulated as shown in Table 1. The problem type can be classified as follows:

- Problems with Continuous Variables (Pc)
  - One dimensional
  - Multi-dimensional and continuously differentiable
  - Multi-dimensional and not continuously differentiable
- Problems with Continuous Variables with inequality constraints (Pcg)
- Problems with Discrete Variables (Pd)
- Problems with Continuous and Discrete Variables (Pcd)
- Problems with Continuous and Discrete Variables with inequality constraints (Pcdg)

Problem Type	Optimisation Algorithm	Abbr.
Pc with n > 1 continuously differentiable	Hybrid algorithm	-
	GPS implementation of the Hooke-Jeeves algorithm	GPSHJ
	Discrete Armijo Gradient	DAG
Pc with n > 1 Not continuously differentiable	Hybrid algorithm	-
	GPS implementation of the Hooke-Jeeves algorithm	GPSHJ
	Particle Swarm Optimisation	PSO
Pcg with n > 1	Hybrid algorithm	-
	GPS implementation of the Hooke-Jeeves algorithm	GPSHJ
Pc with n = 1	Golden Section Interval Division	-
	Fibonacci Division	-
	Parametric	-
Pcg with n = 1	Golden Section Interval Division	-
	Fibonacci Division	-
	Parametric	-
Pd	Particle Swarm Optimisation	PSO
Pcd and Pcdg	Hybrid algorithm	-
	Particle Swarm Optimisation	PSO

Table 1: Algorithms classified by problem type

Apart from the problems with discrete variables (Pd), more than one optimisation algorithm can be used. The flowchart shown in Figure 1 has, therefore, been generated based on the literature to facilitate the selection of the recommended algorithms (Cacabelos et al., 2016). As illustrated in the flowchart for problems with discrete variables (Pd) and for problems with continuous and discrete variables with or without inequality constraints (Pcd, Pcdg), the possible solution(s) can be found directly whereas, for problems with continuous variables, several factors must be taken into consideration. One of the main factors is whether the input parameters are one-dimensional (n=1) or multi-dimensional (n>1). For a problem with continuous variables in one dimension (Pc with n=1) the solution is independent from the constraints whereas, for a problem with continuous variables in multi-dimensions (Pc with n>1), the solution depends both on the constraints and the differentiability of the cost function. The Generalised Pattern Search algorithms with implemented Coordinate Search algorithms are not included as Coordinate Search can only converge to optimal values when the cost function is smooth (Wetter and Wright, 2003). The Nelder and Mead's Simplex algorithm was not included because its usage is not recommended if the cost function has large discontinuities.

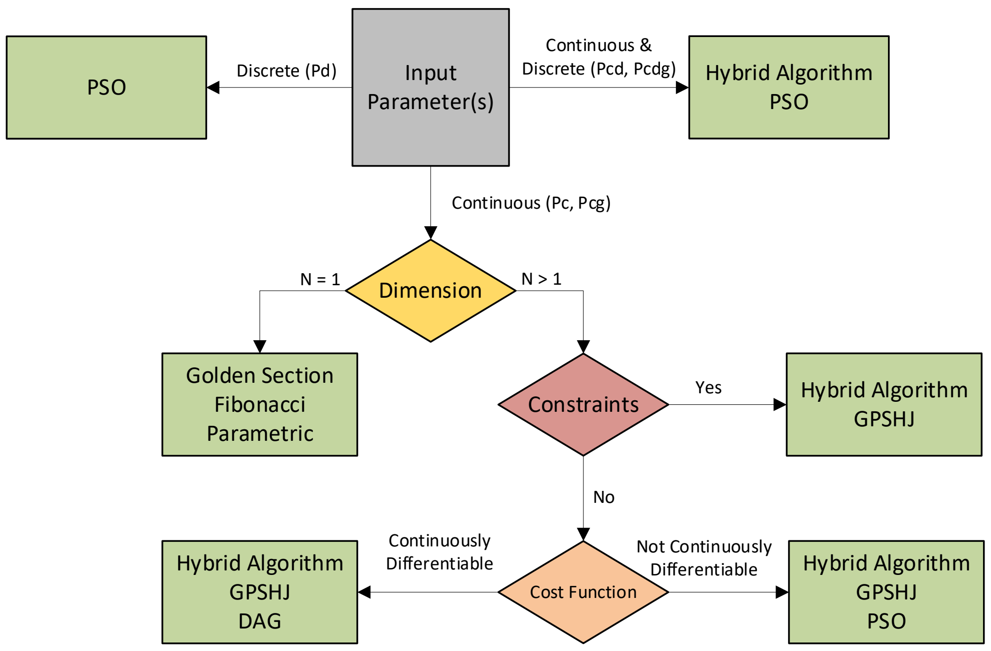

PSO
Discrete (Pd)
Input Parameter(s)
Continuous & Discrete (Pcd, Pcdg)
Hybrid Algorithm PSO
Continuous (Pc, Pcg)
N = 1
Dimension
N > 1
Golden Section Fibonacci Parametric
Constraints
Yes
Hybrid Algorithm GPSHJ
No
Hybrid Algorithm GPSHJ DAG
Continuously Differentiable
Cost Function
Not Continuously Differentiable
Hybrid Algorithm GPSHJ PSO

Figure 1: GenOpt recommended algorithm flowchart

Selection Matrix

For the problems Pc, Pcg, Pcd, and Pcdg, several optimisation algorithms have been recommended and can be used. Selecting the right optimisation algorithm requires a certain level of knowledge and experience. For a user without significant understanding of the optimisation algorithms, the selection process can be a challenging task. Therefore, a selection matrix has been created (Table 2) to facilitate the selection of the optimisation algorithms from the groups of two or three algorithms recommended in Figure 1. The criteria and presented weights for evaluating different available algorithms were adapted from the systematic approach for the selection of optimisation algorithms (Entneret al., 2019). In the algorithm selection matrix, the GenOpt algorithms are stated in the rightmost columns and the user requirements are presented in rows. Each algorithm has been scored based its suitability for each of the set user requirements. It has been evaluated according to the authors' theoretical knowledge and practical experience using a five-point Likert scale, with a score of 1 meaning the algorithm does not fulfil the criterion and 5 meaning it fulfils the criterion.

Key			**Algorithms**													
0 – Not relevant 1 – Optional 2 – Relevant 3 – Required 4 – Essential			Hybrid Algorithm		GPSHJ		DAG		PSO		Golden Section		Fibonacci		Parametric	
Rating	**Weight**	**User Requirement**														
2.6	0.13	Convergence to optimum value	5	0.65	4	0.52	3	0.39	4	0.52	5	4.00	4	0.52	4	0.52
3.4	0.17	Low computational time	2	0.34	4	0.68	3	0.51	3	0.51	5	0.85	5	0.85	2	0.34
0.6	0.03	Generation of alternative designs	5	0.15	5	0.15	4	0.12	4	0.12	4	0.12	3	0.09	4	0.12
2.0	0.10	Ease of use	3	0.30	5	0.50	4	0.40	4	0.40	5	0.50	5	0.50	5	0.50
1.4	0.07	Algorithm customisation	2	0.14	5	0.35	3	0.21	3	0.21	4	0.28	3	0.21	5	0.35
3.4	0.17	Applicability range	5	0.85	4	0.68	3	0.51	4	0.68	4	0.68	3	0.51	4	0.68
3.4	0.17	Low integration effort	3	0.51	3	0.51	4	0.68	4	0.68	5	0.85	5	0.85	3	0.51
2.6	0.13	Comprehensibility	3	0.39	5	0.65	4	0.52	4	0.52	5	0.65	5	0.65	5	0.65
0.6	0.03	Trade-off decision support	5	0.15	3	0.09	4	0.12	4	0.12	4	0.12	3	0.09	4	0.12
20	1	<<< Total Score >>>		3.48		4.13		3.46		3.76		8.05		4.27		3.79

Table 2: Algorithm selection matrix

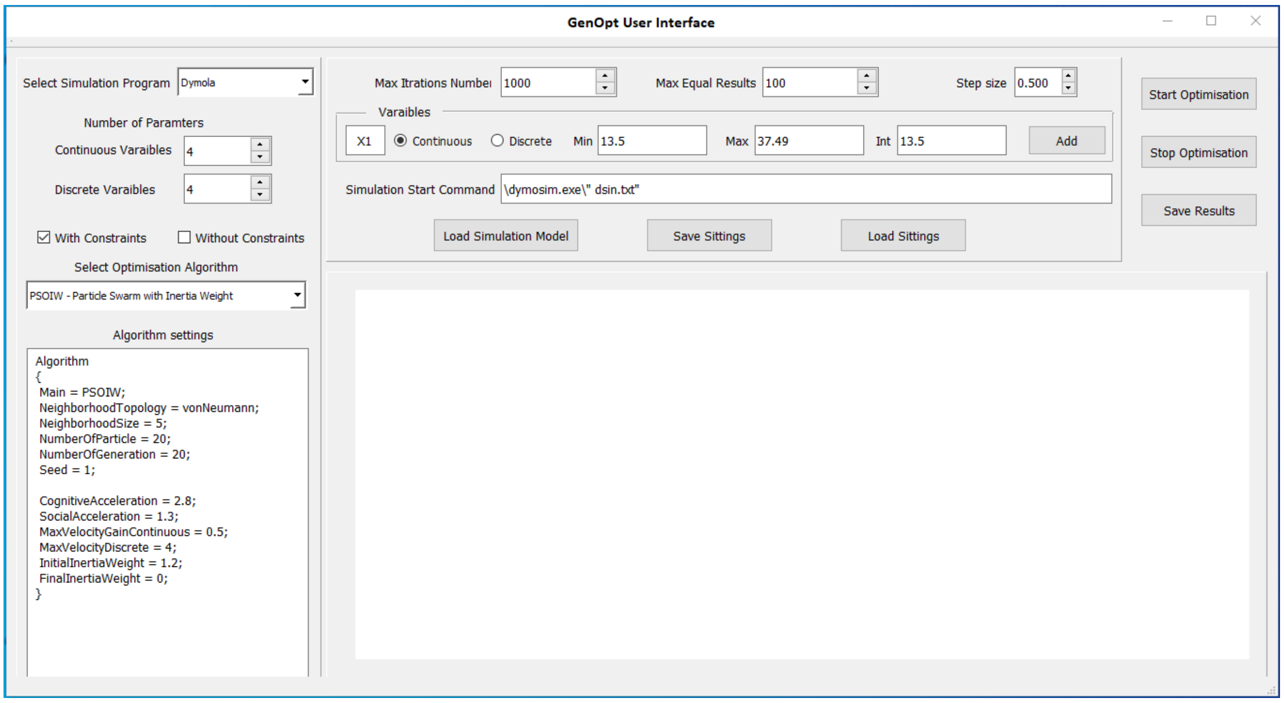
The weight of each criterion is derived from the rating specified by the user (highlighted in yellow) on a scale of 0 (not relevant) to 4 (essential). Once the user finds the suitable algorithms based on the type of problem from Figure 1, the selection matrix in Table 2 allows the user to evaluate such algorithms and identify the recommended choice from the algorithm list based on the overall rating. The overall rating is a sum of scores based on user needs (evaluated on a scale of 0 to 4 and converted to weights) and conducted algorithmic evaluations.

The following definitions of user requirements are considered:

- Convergence to optimum value: The ability to consistently converge toward near-optimal values
- Low computational time: The ability to find acceptable solutions for the user within an acceptable time frame.
- Generation of new design alternatives: The ability to find new design alternatives not generated through small variations of the initial system configuration.
- Ease of use: Minimising effort needed to prepare the algorithm for the problem, including the selection process for the algorithmic parameter values and the user's interaction with the algorithm.
- Algorithm customisation: The ability and the ease of making changes to the algorithm to increase the application range or modify parts of the algorithm.
- Applicability range: The effort needed to apply the algorithm to the whole problem range, including lower or higher dimensional variants of the problem or problems with slightly modified constraints.
- Low integration effort: The effort required to integrate the algorithm into the system and to establish connections with the necessary external tools, such as databases, computing resources, and software libraries.
- Comprehensibility: The ease of understanding the process of solution creation and representation without expert knowledge.
- Trade-off decision support: The ability to investigate various objectives and constraint values with the same or slightly changed algorithm.

GenOpt User Interface

Once the user has decided on the optimisation algorithm, the algorithm section on the command file must be structured in a specified format in order to invoke the algorithm. Invoking the algorithm and specifying optimisation settings, such as parameters, requires coding. To facilitate the process, a GenOpt user interface was developed on C++ as shown in Figure 2. Based on the number of parameters and the constraints, the user interface identifies the type of the optimisation problem and lists the algorithms which can solve it. Once the user selects the algorithm, the user interface automatically uploads the recommended typical values for each parameter.



Select Simulation Program: Dymola

Number of Paramters
Continuous Varaibles: 4
Discrete Varaibles: 4

[✓ With Constraints] Without Constraints

Select Optimisation Algorithm: PSOIW - Particle Swarm with Inertia Weight

Max Itrations Numbei: 1000
Max Equal Results: 100
Step size: 0.500

Varaibles
X1 [✓ Continuous] Discrete Min 13.5 Max 37.49 Int 13.5

Simulation Start Command: \dymosim.exe\" dsin.txt"

Algorithm settings

```
Algorithm
{
 Main = PSOIW;
 NeighborhoodTopology = vonNeumann;
 NeighborhoodSize = 5;
 NumberOfParticle = 20;
 NumberOfGeneration = 20;
 Seed = 1;

 CognitiveAcceleration = 2.8;
 SocialAcceleration = 1.3;
 MaxVelocityGainContinuous = 0.5;
 MaxVelocityDiscrete = 4;
 InitialInertiaWeight = 1.2;
 FinalInertiaWeight = 0;
}
```

Figure 2: Developed user interface

The user can amend the initially set parameters according to their own preference. The settings, including all data displayed on the user interface, can also be saved in a file and can be uploaded if the user wants to repeat the optimisation in the future.

The detailed structure and data exchange between the user interface, the GenOpt program, and the simulation software are illustrated in Figure 3. According to the selected values, the user interface will generate the command file, initialisation file, configuration file and simulation template file to enable the GenOpt to start the optimisation.

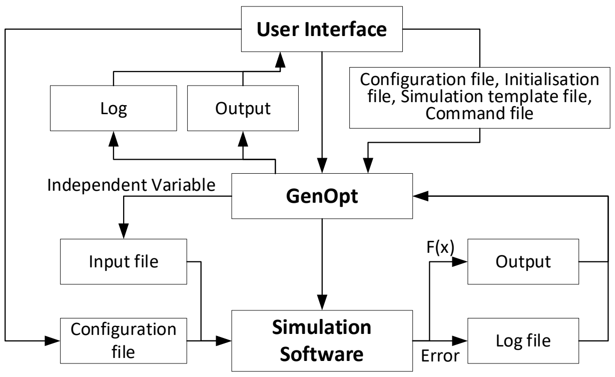


Figure 3: GenOpt user interface architecture

Results

Algorithm Selection Process

The established algorithm selection framework allows the user to identify possible algorithm(s) based on the type of problem, evaluate the algorithms based on the user's own requirements, and integrate the selected algorithm into GenOpt without the need for coding.

Case Study

The introduced GenOpt user interface has been used to solve Unit Commitment (UC) and Economic Load Dispatch (ELD) problems (Khunkitti et al., 2019; Dewangan, Jain and Huddar, 2015). In UC and ELD problems, optimised short-term scheduling of electrical power generation is obtained. Electricity generation and power system providers must meet the varying demand for electricity while minimising the total fuel cost of the generation units over a study period of typically a day. There are two related optimisation problems: UC, which is the process of deciding when and which generation units to start up and shut down, and ELD, the process of deciding the setpoint output of each generation unit at each time-point. UC and ELD problems are subject to many constraints that must be satisfied. A model which represents the objective function of both UC and ELD problems has been derived and coded using C++. The model reads its inputs and writes its outputs to text files. When the optimisation starts, the user interface automatically generates the text files needed to run the GenOpt and calls it to start the optimisation. During the optimisation, GenOpt generates the input parameters and sends it to the UC and ELD model input file and GenOpt launches the model to evaluate the cost function. GenOpt reads the cost function from the UC and ELD output file once it has been evaluated. Based on the value of the output of the model, GenOpt will determine the input parameters for the next run. The process is repeated iteratively and, in each iteration, GenOpt generates a new set of input parameters to the UC and ELD model until a minimum of the cost function is found.

The independent variables of the UC are discrete, the ELD are continuous and their numbers are equal to and greater than one. The problem is subject to a number of equality and inequality constraints. Then, only algorithms which can solve problems with multiple discrete and continuous variables (n>1) and that accept constraints, can be used. Based on the selections and data entered into the user interface shown in Figure 2, the interface will provide the user with a list of the relevant GenOpt optimisation algorithms that can solve this problem. Accordingly, either PSO or GPSPSOCCHJ (hybrid) can be used. The user can then use the proposed selection matrix table to select the most appropriate algorithm out of these two. In this problem, the preferences are set as shown in Table 3.

User Requirement	Rating
Convergence to optimum value	4
Low computational time	3
Generation of alternative designs	1
Ease of use	2
Algorithm customisation	0
Applicability range	2
Low integration effort	0
Comprehensibility	1
Trade-off decision support	3

Table 3: Selection preferences

By applying these ratings to Table 2, the score for the hybrid algorithm (GPSPSOCCHJ) was 4.06 and for PSO was 3.81. After the user selects the preferred optimisation algorithm, the recommended values of the algorithm's parameters and settings will be displayed. These values can be easily amended through the user interface. Evolutionary algorithms, such as the PSO and hybrid algorithms, are popular and effective optimisation algorithms used in building design optimisation (Machairas, Tsangrassoulis and Axarli, 2014). Therefore, the same procedure applies to the selection, evaluation, and optimisation of cost functions generated by building simulation software.

Discussion

Algorithm(s) Selection

From the flowchart in Figure 1, it can be determined that for Problems with Discrete Variables (Pd) only the Particle Swarm Optimization (PSO) algorithm can be used. In contrast, for Problems with Continuous Variables (Pc, Pcg) and Problems with Continuous and Discrete Variables (Pcd, Pcdg) several algorithmic options are available. The flowchart (Figure 1) is effective in selecting suitable algorithms from those already integrated in GenOpt. As GenOpt allows users to implement their own algorithms, the introduction of the user's own algorithm will increase the possible solutions for certain types of problems, therefore making the use of the algorithm selection flowchart impractical.

Selection Matrix

The selection matrix allows the user to evaluate the algorithms found in Figure 1 based on the user's needs. The evaluation is based on two main elements: the user requirement and the ability of the algorithm to fulfil that requirement. Although the selection matrix has been designed specifically with GenOpt in mind, the concept can be applied to the selection of algorithms for any optimisation program that consists of the same or similar optimisation algorithms. When the UC and ELD problems (case study) are evaluated using the established weightage and the user specified weightage, the score changes and, therefore, the preferred solution changes, as shown in Table 4.

	Algorithms	
	Hybrid	**PSO**
Established Weightage	3.48	3.76
User own Weightage	4.06	3.81

Table 4: Score comparison

The pre-established weightage prioritises low computational time, applicability range, and low integration effort, whereas the user's own weightage focuses largely on the convergence to an optimum value. This variation affects the suitability of the algorithm. The scale range used to evaluate the algorithm range is sufficient to distinguish the suitability of the different optimisation algorithms, but having a larger scale range could increase the precision and overall accuracy. To validate the developed approach, the total cost and the average execution time was analysed by varying the demand for electricity eight times, as illustrated in Table 5. For the same number of iterations and similar settings, the GPSPSOCCHJ was able to achieve a lower fuel cost but a much higher average execution time when compared to the PSO.

	Demand for Electricity	**Optimal Fuel Cost**	
		GPSPSOCCHJ	PSO
1	177	2392.17	2577.39
2	507	5599.16	6337.52
3	650	7530.05	8076.81
4	800	9291.55	9731.17
5	989	116960.00	454359.00
6	939	11058.00	11403.60
7	776	8925.48	9921.67
8	355	4474.11	4831.72
Total Cost		166230.50	507238.88
Average Execution Time		755252ms	202351ms

Table 5: Results for eight time slots

GenOpt User Interface

The GenOpt user interface combines the GenOpt optimisation program with both simulation programs and other optimisation algorithms. The user interface presented in Figure 2 enables the user to input information required to optimise the cost function. The default values of all control parameters are provided, and they can be easily changed. The interface consists of four main panels: a selection panel to select the optimisation program and algorithm, an algorithm control panel to adjust the algorithm parameters, an illustration panel to display the set points of all units stacked at each time slot and a cost panel displaying the total cost at each time slot. Additionally, there are option buttons to load the data required for the optimisation, to start the optimisation, and stop the optimisation.

Conclusion

GenOpt can effectively perform optimisation of non-linear problems given the correct optimisation algorithm is chosen, and the right variables and parameters are inserted. The described algorithm selection framework allows the user to take the most appropriate and effective approach and acts as a decision support system. Although algorithm(s) selection and evaluation criterion concepts are implemented from earlier work, the novelty of this research lies in the development of a user-informed decision-making process and the development of a valuable new tool. The developed user interface allows the users to insert and amend algorithmic variables in an interactive, user-friendly environment without the need for coding. This enhances the overall simulation results by enabling the user to better understand the simulation model behaviour, the optimisation algorithm, and its variables. Optimisation is vital for a variety of engineering systems, but not all system operators and professionals have the necessary coding skills to use generic optimisation software, such as GenOpt. Therefore, having a user-friendly interface, such as the one developed, will allow future GenOpt users to apply it more effectively, potentially widening applicability and stakeholder audience. The established process of evaluating and implementing optimisation algorithms in GenOpt using the developed tool and selection matrix contributes to the knowledge of performing optimisation. Further work will include the development of a rule-based approach to fine-tune the optimisation algorithm's parameters in relation to the problem type.

Acknowledgement

This work is partly funded by EU Horizon 2020 eDREAM project (01/01/2018-31/12/2020) under grant agreement No. 774478. The authors wish to acknowledge the European Commission for their support.

References

Antoniou, A. and Lu, W. (2007) *Practical Optimization.* Boston: Springer.

Burke, E.K. *et al.* (2013) 'Hyper-heuristics: a survey of the state of the art', *Journal of the Operational Research Society,* 64(12), pp. 1695-1724. doi: 10.1057/jors.2013.71.

Cacabelos, A. *et al.* (2016) 'Integration of the free software GenOpt for a thermal engineering course', *Computer Applications in Engineering*

Education, 24(3), pp. 356-364. doi: 10.1002/cae.21713.

Crawley, D.B. *et al.* (2001) 'EnergyPlus: creating a new-generation building energy simulation program', *Energy & Buildings,* 33(4), pp. 319-331. doi: 10.1016/S0378-7788(00)00114-6.

Dewangan, S.K., Jain, A. and Huddar, A.P. (2015) 'A Traditional Approach to Solve Economic Load Dispatch Problem Considering the Generator Constraints', *IOSR Journal of Electrical and Electronics Engineering*, 10(2), pp. 27-32.

Dey, N. (2017) *Advancements in Applied Metaheuristic Computing.* Hershey: IGI Global.

Drake, J.H. *et al.* (2020) 'Recent advances in selection hyper-heuristics', *European Journal of Operational Research,* 285(2), pp. 405-428. doi: 10.1016/j.ejor.2019.07.073.

Gabbar, H. (2016) *Smart Energy Grid Engineering.* 1st edn. San Diego, CA, USA: Elsevier Science.

Khunkitti, S. et al. (2019) 'An Improved DA-PSO Optimization Approach for Unit Commitment Problem', *Energies*, 12(12), pp. 2335. doi: 10.3390/en12122335.

Leprêtre, F. *et al.* (2019a) 'Fitness landscapes analysis and adaptive algorithms design for traffic lights optimisation on SIALAC benchmark', *Applied Soft Computing Journal,* 85, pp. 105869. doi: 10.1016/j.asoc.2019.105869.

Leprêtre, F. *et al.* (2019b) 'Fitness landscapes analysis and adaptive algorithms design for traffic lights optimisation on SIALAC benchmark', *Applied Soft Computing Journal,* 85, pp. 105869. doi: 10.1016/j.asoc.2019.105869.

Machairas, V., Tsangrassoulis, A. and Axarli, K. (2014) 'Algorithms for optimisation of building design: A review', *Renewable and Sustainable Energy Reviews*, 31, pp. 101-112.

Marks, P., Gerrits, L. and Marx, J. (2019) 'How to use fitness landscape models for the analysis of collective decision-making: a case of theory-transfer and its limitations', *Biology & Philosophy,* 34(1), pp. 1-15. doi: 10.1007/s10539-018-9669-4.

Osman, I.H. and Kelly, J.P. (1996) *Meta-heuristics: an overview.*

Waibel, C. *et al.* (2019) 'A comparison of building energy optimisation problems and mathematical test functions using static fitness landscape analysis', *Journal of Building Performance Simulation,* 12(6), pp. 789-811. doi: 10.1080/19401493.2019.1671897.

Wetter, M. (2000) 'Design optimisation with GenOpt', *Building Energy Simulation User News,* 21(19-28).

Wetter, M. (2001) 'GenOpt-A generic optimisation program', *Seventh International IBPSA Conference, Rio de Janeiro.*

Wetter, M. (2016) GenOpt(R), generic optimisation program, User Manual, Version 3.1.1. Berkeley, CA: University of California: Lawrence Berkeley National Laboratory.

Wetter, M. and Wright, J. (2003) 'Comparison of a generalised pattern search and a genetic algorithm optimisation method', *Proceedings of the 8-th IBPSA Conference*.

Wortmann, T. and Nannicini, G. (2017) 'Introduction to architectural design optimisation', in 'Introduction to architectural design optimization'*City Networks.* Springer, pp. 259-278.