Optimization Algorithms for Energy Building Applications

S Tsala¹, I P Koronaki¹

¹Laboratory of Applied Thermodynamics, School of Mechanical Engineering, Thermal Engineering Section National Technical University of Athens, Heroon Polytechniou 9, Zografou Campus 15780, Athens, Greece

Corresponding author’s e-mail: koronaki@central.ntua.gr

Abstract. The creation of the structured environment affects the environment in all three phases of the construction and the demolition of buildings. The appropriate design of buildings requires full identification of the interaction between the environment and the structured environment. Energy efficiency is a mandatory requirement and integral part of green and sustainable buildings. Energy efficient design optimization is both a design philosophy and a practical technique which has been proposed and used by architects and other professionals for several decades, especially in recent years. Bioclimatic construction has been an important tool for improving the construction of buildings in the past few decades. Energy-saving options in buildings include the use of appropriate materials, exposures and the use of alternative power generation systems. In the present work, the methods and tools used for building design optimization are displayed in an effort to explore the reasoning behind their selection, to present their abilities and performance issues, and to identify the key characteristics of their future versions. In this essay, several algorithms are provided for optimizing problems which are proposed for building construction. Depending on whether limitations are given and on the general form of the optimization problem, one could choose the optimization algorithm which suits each case.

1. Introduction
Optimization is a process which seeks to identify the best solution for a given problem. Any optimization problem can be considered as a problem of mapping from decision-making to objective [1,2]. Solutions to an optimization problem are feasible when they meet all the requirements defined by the problem, while the optimal solution can minimize the cost of the process or maximize the effect of the solution to the problem. Typically, optimization problems are used to solve various algorithms. The algorithm is a well-defined computational process which has input data of one or more variables and produces a set of values as output values. Therefore, an algorithm is a computational process which converts input data into output results [3].

The field of building construction is considered to be an interdisciplinary area as multidimensional designing skills are necessary. One pillar of this ecosystem of expertise is computational engineering which can contribute to the amelioration of designing methods for the energy consumption of buildings. Since 2000, when Hien, Poh, & Feriadi(2000)[4] postulated that real life architects cannot use dedicated algorithms and software for building optimization due to time and economic pressure, things have dramatically changed. The relevant literature has been enriched and a significant part of research is in progress aiming at the formulation of algorithms which would optimize the energy performance of buildings.
For example, a crucial part for energy consumption is lighting. Glaser et al. (2004)\[5\] attempted to develop an algorithm for building lighting optimization and presented its implementation to “LightSketch”, a dedicated software to lighting design. On the other hand, Castro-Lacouture, Sefair, Flórez & Medaglia(2009)\[6\] realized that green buildings need to be not only energy efficient while in full function but also built with the proper technologies and materials as inappropriate material selection can cause energy problems. Furthermore, Baglivo, Congedo, Fazio & Laforgia(2014)\[7\] focused on the amelioration of buildings located in the Mediterranean. Their study was based on the implementation of a multi-objective approach in order to optimize the efficiency of external walls. The algorithm implementation included multiple factors such as thermal transmittance, periodic thermal transmittance, decrement factor, time shift, areal heat capacity, thermal admittance, surface mass, thickness.

Yu, Li, Jia, Zhang & Wang (2015)\[8\] introduced a multi-objective genetic algorithm based on the application of an improved multi-objective genetic algorithm (NSGA-II), a simulation-based improved Back propagation network, and the Pareto solution to obtain a building design multi-objective optimization model. According to the researchers, the application of the specific algorithm would help building designers detect the cross-section between thermal comfort and energy consumption. Touloupaki and Theodosiou(2017)\[9\] also focused on Multi-Objective Optimization Algorithms integrating evolutionary algorithms and energy simulation through Grasshopper for Rhinoceros 3d. On the other hand, it is well understood that the implementation of such algorithms presents a high risk of uncertainty. Bamdad, Cholette, Guan & Bell (2018)\[10\] introduced a modified version of the Ant Colony Optimization algorithm for Mixed Variables (ACOMV-M) in order to deal with problems arising from such uncertainty.

The issue of the high level of uncertainty presented by each algorithm has become the focus of research. Numerous studies attempt to compare and spot disadvantages among proposed solutions. Si, Tian, Jin, Zhou & Shi, Shi (2019)\[11\] tried to present the problematic issues related to optimization algorithms and analyze the factors affecting their effectiveness. Their study focuses on the comparison of Armijo gradient, Hooke-Jeeves, particle swarm optimization with constriction coefficient (PSOCC) and particle swarm optimization with inertia weight (PSOIW) and aims at determining weaknesses and assisting designers with their work. Accordingly, Kheiri(2018)\[12\] attempted to shed light on existing tools and algorithms.

The most important optimization algorithms will be analyzed and compared in this paper.

2. Local Optimization Algorithms

A local algorithm is a distributed algorithm which runs on a fixed number of synchronized circles, regardless of the number of nodes in the network. In other words, the output of a node in a local algorithm is a function of the input that is available in a fixed radius neighborhood of the node. Therefore, the output of a node in a local algorithm is a function of the available input in a neighboring fixed-radius region of the node \[13\].

In general, local optimization methods are based on the iterative process of finding a solution in which a simple design point alternates in the different steps of the iterative process until the optimal solution is determined. Consequently, local optimization algorithms require the expression of the problem through an objective function.

2.1. Golden Section algorithm

The Golden section method is used to find the maximum or minimum of a monotonous function, that is, a function that contains only the minimum or maximum space \([a, b]\).

This algorithm consists of the following steps \[14\]:

**Step 1:** Two points \(a\) and \(b\) are defined between which the maximum (or minimum) of the function \(f(x)\) is known to exist. Then, two intermediate points \(x_1\) and \(x_2\) are defined such that \(x_1 = a + d\) and \(x_2 = b - d\) where:

\[
d = \frac{\sqrt{5} - 1}{2} \times (b - a)
\]  

(1)

**Step 2:** Values for the function \(f(x_1)\) and \(f(x_2)\) are estimated. If \(f(x_1) > f(x_2)\) then new \(a, x_1, x_2\) and \(b\) are defined so that:
The exact form of the algorithm used is shown in the figure below:  

In general, this algorithm has been widely studied for photovoltaic systems and battery performance. The specific algorithm is preferred because of the rapid convergence and the noise resistance.

In recent work [15] the Golden Search algorithm was used to estimate the performance of moving photovoltaics using battery, current and voltage characteristics and to examine the effect of shading on battery performance. The specific algorithm is preferred because of the rapid convergence and the noise resistance. In general, this algorithm has been widely studied for photovoltaic systems in buildings, assessing the parameters which may affect their performance. Gayarthi & Ezhilarasi [16] also used this algorithm with a system of inverter in photovoltaic systems to determine their performance and proved that the voltage ratio is not linear and that the maximum power exists for specific current and voltage values. The maximum power varies according to the atmospheric conditions and the radiation that the system receives. In addition, by using the algorithm it was proved that the inverter performance reaches 90%.

The above algorithm is satisfactory when it is possible to determine a and b. Otherwise the algorithm changes in a stricter way, in order to determine the required primary guesses for the lower and upper limits required and to find only one intermediate point (instead of two). In this case, the algorithm becomes[13]:

**Step 1:** For a small incremental \( \delta \) size in space, e.g. \( \delta = 0.05 \), \( f \) is chosen as the smallest integer such as to apply:

\[
g \left( \sum_{v=0}^{j} \delta(1, 1.618)^v \right) g \left( \sum_{v=0}^{j-1} \delta(1, 1.618)^v \right)
\]

The upper and lower limit of \( a^i \) is \( a_u = \sum_{v=0}^{L\ell \bar{a}} \delta (1.618)^v \) and \( a_L = \sum_{v=0}^{L-2} \delta (1.618)^v \)

**Step 2:** Calculate \( g(a_u) \) where \( a_u = a_u^* + 0.382(a_u^* - a_L) \) and \( a_b = a_u^* - 0.182(a_u^* - a_L) \). Given that \( a^* = \sum_{v=0}^{j} \delta (1.618)^v \) the function \( g(a_u) \) is known.

**Step 3:** Compare \( g(a_u) \) and \( g(a_b) \) follow steps 4, 5 and 6.

**Step 4:** If \( g(a_u) < g(a_b) \) then \( a_u \leq a^i \leq a_b \) from the choice of \( a_u \) and \( a_b \) the new points \( \bar{a}_L = a_u \) and \( \bar{a}_U = a_b \) have \( \bar{a}_U = a_u \). In this case it is calculated \( g(\bar{a}_U) \) where \( \bar{a}_U = a_u + 0.382(\bar{a}_U + \bar{a}_U) \) and proceeds to step 7.

**Step 5:** If \( g(a_u) > g(a_b) \) then \( a_u \leq a^i \leq a_u \) as in step 4 is set \( \bar{a}_L = a_u \) and \( \bar{a}_U = a_b \). Calculated \( g(\bar{a}_U) \) where \( \bar{a}_U = a_u + 0.618(\bar{a}_U + \bar{a}_U) \) and followed by step 7.

**Step 6:** If \( g(a_u) = g(a_b) \) set \( a_u = a_u \) and \( a_b = a_b \) and return to step 2.

**Step 7:** If \( \bar{a}_U - \bar{a}_L \) is too small set \( a^i = \frac{1}{2} (\bar{a}_U + \bar{a}_L) \) and stop the process. Otherwise, the process returns to step 3 after the bar is switched off \( \bar{a}_U = \bar{a}_U, \bar{a}_L = \bar{a}_L \), \( \bar{a}_U \) and \( \bar{a}_U \).
We will give an example of the Golden Section method. Let us minimize the function 
\[ f(x) = x^4 - 14x^3 + 60x^2 - 70x \], in the space \([0, 2]\), with 0.3 accuracy.

First of all, we define \(a=0\), \(b=2\) and \(r = (\sqrt{5} - 1)/2 \approx 0.618\).

**Step 1:** It is 
\[ x_1 = a + r(b - a) = 0.764 \] and \( x_2 = b - r(b - a) = 1.236 \). The \(f\)-values of \( x_1 \) and \( x_2 \) are \( f(x_1) = -24.36 \) and \( f(x_2) = -18.96 \). Since \( f(x_1) < f(x_2) \), the new search space will be the \([a, x_2]\).

**Step 2:** We choose \( x_3 = a + r(x_2 - a) = 0.472 \) and \( x_4 = x_3 \), hence \( f(x_3) = -21.10 \) and \( f(x_4) = -24.36 \). Since \( f(x_3) > f(x_4) \), the new search space shall be \([x_3, x_2]\).

**Step 3:** We choose \( x_5 = x_4 \) and \( x_6 = x_3 + (1 - r)(x_4 - x_3) = 0.944 \), hence \( f(x_5) = -24.36 \) and \( f(x_6) = -23.59 \). Since \( f(x_5) < f(x_6) \), the new search space shall be \([x_3, x_5]\).

**Step 4:** For \( x_7 = x_5 + r(x_7 - x_5) = 0.653 \) and \( x_8 = x_5 \), we have \( f(x_7) = -23.84 \) and \( f(x_8) = -24.36 \). Since \( f(x_7) > f(x_8) \), the new search space shall be \([x_7, x_2]\).

The algorithm terminates here, because \( x_7-x_8=0.292 \leq 0.3 \). So the answer is that the minimum \(x^*\) is at the space \([0.653, 0.944]\).
2.2. Fibonacci Search

In computer science, Fibonacci Search is used to search into a classified array by using a divide-and-conquer algorithm that reduces the number of possible locations using Fibonacci numbers. The specific method examines only those locations which have the smallest dispersion, reducing the overall runtime of the algorithm [17].

We consider \( k \) as an element in \( F \), which is the Fibonacci numbers’ array, and \( n = F_m \) is the size of the array. If \( n \) is not a Fibonacci number, \( F_m \) is considered to be the smallest number in \( F \) that is greater than \( n \).

The Fibonacci numbers are defined by the relationship \( F_{k+2} = F_{k+1} + F_k \) when \( k \geq 0, F_0 = 0 \) and \( F_1 = 1 \). Given a table of entries \( R_1, R_2, \ldots, R_n \) whose keys increase in that order: \( K_1 < K_2 < \cdots < K_n \), the Fibonacci algorithm searches for a given statement \( K \). It is assumed that \( n + 1 = F_{k+1} \). The Fibonacci search algorithm is described in the following steps:

**Step 1:** Set: \( F_k \rightarrow i, F_{k-1} \rightarrow p \) and \( q \rightarrow F_{k-2} \). In the algorithm, \( p \) and \( q \) are sequential numbers of Fibonacci.

**Step 2:** A comparison is made. If \( k < k_i \), then step 3 follows, if \( k > k_i \), step 4 follows and if \( k = k_i \) the algorithm terminates successfully.

**Step 3:** Reduce. If \( q = 0 \), the algorithm terminates without completing. Otherwise, \((p, q, p - q) \rightarrow (i, p, q)\) is set. In this way, \( p \) and \( q \) move a position backwards in the Fibonacci series and the algorithm returns to step 2.

**Step 4:** Increase \( i \). If \( p = 1 \), the algorithm terminates without completing. In the opposite case, set \((i + q, p - q, 2q - p) \rightarrow (i, p, q)\) namely \( p \) and \( q \) are moved two positions backwards in the Fibonacci series and the algorithm returns to step 2.

Lin and Ma [18] have optimized phase shifting of building enhancing materials with integrated photovoltaic systems and concrete photovoltaic panels that are built into the building shell to increase the thermal mass of the building while the heated air generated by the photovoltaic system was used to heat space. The aim of the simulation was to maximize the signal / noise ratio of the thermal efficiency boosting factor of buildings. The optimized factors were the air flow rate, the type of material phase transformation, the thickness of the material and the insulation thickness of the wall. The combination of the Taguchi method and the Fibonacci method was used for optimization.
The following symbols are associated with the following chart:

- $f^k$ → factor interval at current iteration
- $x_i \rightarrow f^k$ continuous factor
- $z_j \rightarrow j^{th}$ discrete factor
- $\% C \rightarrow$ percentage contribution
- $a_k^l \rightarrow$ Lower limit of continuous factor search range at current iteration
- $b_k^u \rightarrow$ Upper limit of continuous factor search range at current iteration
- $a_k^l, b_k^u \rightarrow$ factor interval candidates generated by Fibonacci search at current iteration

Figure 2: Outline of the optimization methodology
Figure 3. The Fibonacci Taguchi optimization method used by Li and Ma (2016)
The optimization results were based on a typical Australian home and specified airflow values, insulation thickness, and phase change factors of the materials.

2.3. The Fletcher-Reeves method

The Fletcher-Reeves method is a non-linear conjugate gradient method mainly used for nonlinear optimization problems [19]. The Fletcher-Reeves method is used to solve the unconstrained optimization problem: minimize $f(x)$, $x \in \mathbb{R}^n$ using a series of searches of the form: $x^{k+1} = x^k + a^k d^k$ after first estimating $x^k$. The Fletcher-Reeves variable in the conjugate tilt algorithm generates the $d_{k+1}$ of the relations:

$$d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$$  \hspace{1cm} (11)  

$$\beta_k = \left(\frac{\|\nabla f(x_{k+1})\|_2}{\|\nabla f(x_k)\|_2}\right)^2$$  \hspace{1cm} (12)

The Fletcher-Reeves method can be used to estimate the boundary heat flows in a building and to solve the reverse convection problem without having any information about the previous heat flow [20].

![Figure 4_Example for Fletcher-Reeves process](image-url)
Figure 5_Solving the Fletcher Reeves Algorithm to solve the inverse conundrum problem in buildings. (Zhao et al 2009)
2.4. Successive repeat algorithm without limitations
Suppose that one wants to minimize the real function \( f: X \rightarrow R \) that is likely to be subjected to constraints of the independent variable through the sequential repeat algorithm without restrictions. The main problem in convex programming is to minimize a convex function \( f: R^j \rightarrow R \) for which \( h_i(x) \leq 0 \) for \( i = 1, 2, \ldots, I \) and \( h_i \) are convex. If the problem is consistent, that is, there is \( x \) that satisfies the condition \( h_i(x) \leq 0 \), then the necessary and sufficient condition for solving the problem is that there exists \( \lambda_i * \geq 0 \) for which \( \lambda_i * h_i *( x^* ) \) for each \( i \) and \( x^* \) minimizes the function \( f(x)+\sum_{i}^{I}\lambda_i h_i(x^*) \)

In the \( k \) step of the sequential repetitive process without limitations, the function \( G_k(x) = f(x) + g_k(x) \) is minimized to find \( x^k \) while \( g_k(x) \) is a function selected by the user. The goal is to find a series \( \{x^k\} \) that converges to solution \( x \) of the original problem. Otherwise, the values of the function \( f(x^k) \) subjected to fewest restrictions are searched for.

2.5. Powell Algorithm
Powell discovered an algorithm for finding a local minimum of a function. Having a function with \( N \) parameters, an initial parameter guess and \( N \) search vectors should be provided. In this way, it is possible to write the new parameter as a linear combination of all search vectors. After that, a new search direction is placed there where there is the search direction that gives the higher impact. The algorithm is repeated until convergence [21].

**Step 1:** Set the original \( p_0 \) position.
**Step 2:** For \( i = 1, 2, \ldots, N \), the \( p_{i-1} \) moves to the minimum \( u_i \) direction and this point is defined as \( p_i \).
**Step 3:** For \( i = 1, 2, \ldots, N-1 \) set \( u_{i+1} \rightarrow u_i \)
**Step 4:** Set \( p_N = p_0 \rightarrow u_N \)
**Step 5:** \( p_N \) moves to a minimum of \( u_N \) direction and this point is defined as \( p_0 \).

Powell showed that, for a square format, the iterations of the above basic procedure produce a set of \( u_i \) directions whose last \( k \) members are mutually conjugated. Therefore, the \( N \) iterations of the basic procedure, which correspond to \( N(N + 1) \) line minimization, minimize exactly a square form. The disadvantage of this algorithm is that at each stage \( u_N \) in relation to \( p_N-p_0 \) tends to produce sets of directions that "fold over the others", resulting in linear dependencies. When this happens, the process finds the minimum of \( f \) only on a subset of the total space, ie it leads to incorrect results.

The example given is based on the following Powell method algorithm. It better illustrates the process followed in the example.

**Step 1:** Define \( p_0 = X_i \)
**Step 2:** For \( i = 1, 2, \ldots, N \), find the value of \( a_i \) that minimizes \( f(P_{i-1} + a_i U_i) \) and set \( P_i = (P_{i-1} + a_i U_i) \).
**Step 3:** Set \( j = j + 1 \)
**Step 4:** Set \( U_k = U_{k-1} \) for \( k = 1, 2, \ldots, N-1 \). Set \( U_N = p_n - p_0 \)
**Step 5:** Find the value of \( a_i \) that minimizes \( f(P_0 + a U_N) \) and set \( X_i = (P_0 + a U_N) \)
**Step 6:** Repeat Steps 1 through 5.

We will see an example of Powell’s Method at the function \( f(x, y) = \cos(x) + \sin(y) \), using the initial point \( X_0 = (5.5, 2) \).

Firstly, \( U = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \) and \( p_0 = X_0 = (5.5, 2) \). For \( i = 1 \), we have in the function
\[
f(P_0 + a U_1) = f((5.5, 2) + a_i(1, 0)) = f(5.5 + a_i, 2) = \cos(5.5 + a_i) + \sin(2)
\]
minimum at \( a_1 = -2.3584042 \) So \( P_1 = (3.1415958, 2) \). For \( i = 2 \), the function
\[
f(P_0 + a U_1) = f((5.5, 2) + a_i(1, 0)) = f(5.5 + a_i, 2) = \cos(5.5 + a_i) + \sin(2)
\]
has a minimum at \( a_2 = 2.7123803 \). So, \( P_2 = (3.1415958, 4.7123803) \). Set \( U_2 = (P_2 - P_0) \)’ and
\[
U = \begin{bmatrix} 0 & -2.3584042 \\ 2.7123803 \end{bmatrix}.
\]
In some embodiments, the vertices of a repeat of the Nelder-Mead algorithm consist of the following steps:

1. Construction of the original simplex $S$.
2. Repeat the following steps to satisfy the shutdown test: The shutdown information is calculated and if it is not satisfied, then the original simplex is converted.
3. Return to the best top of the current simplex and correlate the value of the function.

### Home Simplex

The original Simplex $S$ is usually constructed by creating $n + 1$ vertices $x_0, \ldots, x_n$, around a given entry point $x_{\text{in}} \in \mathbb{R}^n$. In practice, the most common option is $x_0 = x_{\text{in}}$ to allow the algorithm to restart correctly. The remaining vertices $n$ are produced to obtain one of the two patterns of simplex.

- S is rectangular on the $x_o$ based on the coordinate axes or:
  $$x_j = x_0 + h_j e_j, \quad j = 1, 2, \ldots, n$$
- S is a normal simplex, where all edges have the same specified length.

### Simplex Transformation algorithm

A repeat of the Nelder-Mead method consists of the following three steps.

- Layout: Determine the $h, s, 1$ of the worst, second worst, and optimal peaks, respectively, in the current simplex operation.
  $$F_h = \max f_j, \quad F_B = \max_{j \neq h} f_j, \quad F_1 = \min_{j \neq h} f_j$$
  In some embodiments, the vertices of $S$ are arranged relative to the operating values to satisfy:
  $$f_0 \leq f_1 \leq \cdots \leq f_{n-1} \leq f_n$$
  Then $1 = 0, s = n - 1$ and $h = n$.
- Centroid: Calculate the centroid $c$ of the best side that is considered the opposite of the worst peak $x_n$
  $$c = \frac{1}{n} \sum_{j \neq h} x_j$$

The function

$$f(P_0 + aU_j) = f (3.1848261, 4.6626615) + a(0, 1) = f(3.1848261, 46626615 + a_1) = \cos(3.1848261) + \sin(4.6626615 + a_1)$$

Has a minimum at $a_1=0.0497117$. So $P_{1} = (3.1848261, 4.7123732)$. For $i=2$, the function

$$f(P_0 + aU_j) = f (3.1848261, 4.7123732) + a_2(-2.358402, 2.7123803) = f(3.1848261 - 2.358402a_2, 4.7123732 + 2.7123809a_2) = \cos(3.1848261 - 2.358402a_2) + \sin(4.7123732 + 2.7123809a_2)$$

Has a minimum at $a_2=0.0078820$. So $P_{2} = (3.1662373, 4.7337521)$. Set $U_2 = (P_2 - P_0)$ and

$$U = \begin{bmatrix} -2.3584042 \\ 2.7123803 \end{bmatrix}$$

The function

$$f(P_0 + aU_j) = f (3.1848261, 4.6626615) + a(-0.00185889, 0.0710906) = f(3.1848261 - 0.00185889a_1, 4.6626615 + 0.0710906a_1)$$

has a minimum at $a_1=0.8035684$. So $X_2 = (3.1698887, 4.7197876)$. In the end, the function $f(x, y) = \cos(x) + \sin(y)$, has a relative minimum at $P = (\pi, \frac{3\pi}{2})$. 

### 2.6. Nelder-Mead algorithm

The Nelder-Mead algorithm or the downhill simplex algorithm, originally published in 1965 (Nelder and Mead, 1965), is one of the best-known algorithms for multidimensional, unrestricted optimization without the use of derivatives.

Although the method is relatively simple, it can be accomplished in various ways. In addition to some small computational details in the basic algorithm, the main difference between the different forms of implementation is found in the construction of the original Simplex (an optimization method, perhaps the most basic) and in the selection of the convergence or termination tests used in the algorithm. The generic algorithm consists of the following steps [22]:

- Construction of the original simplex $S$.
- Repeat the following steps to satisfy the shutdown test: The shutdown information is calculated and if it is not satisfied, then the original simplex is converted.
- Return to the best top of the current simplex and correlate the value of the function.
IOP Conf. Series: Materials Science and Engineering 1037 (2021) 012040

• Transformation: The new Simplex is calculated. Initially, only the worst \( x_n \) top is replaced with a better spot using reflection, swelling or shyness in relation to the best side. All test points are in the line defined by \( x_n \) and \( c \) and at most two of them are calculated in one iteration. If this succeeds, the acceptable point becomes the new peak of simplex. If this fails, the simplex is shrunk to the best top \( x_1 \). In this case, new peaks \( n \) are calculated.

The transformations of simplex in the Nelder-Mead method are determined by four parameters: \( \alpha \) for reflection, \( \beta \) for contraction, \( \gamma \) for swelling and \( \delta \) for contraction which must meet the conditions: \( \alpha > 0, 0 < \beta < 1, \gamma > 1, 0 < \delta < 1 \).

- In most applications the values used are \( 1, \beta = \frac{1}{2}, \gamma = 2 \) and \( \delta = \frac{1}{2} \).

- **Reflection**: Reflex points are calculated: \( x_r = c + a(c - x_n) \) and \( f_r = f(x_r) \). If \( f_1 < f_r < f_3 \) is accepted the \( x_r \) and terminates the iterative process.

- **Swelling**: If \( f_e < f_1 \) calculates the bulking point \( x_e = c + \gamma(x_r - c) \) and \( f_e = f(x_e) \). If \( f_e < f_r \) is accepted \( x_e \) and the iterative process stops. Otherwise, \( x_r \) is accepted and the process stops.

- **Contraction**: If \( f_r > f_3 \) computes the point inserted \( x_n \) using the best point between the \( x_h \) and \( x_r \).

- **External**: If \( f_3 \leq f_r < f_1 \) computes \( x_c = c + \beta(x_r - c) \) and \( f_c = f(x_c) \).

- **Internal**: If \( f_r \leq f_h \), \( x_c = c + \beta(x_h - c) \) and \( f_c = f(x_c) \) are computed.

- **Shrinking**: New peaks \( x_n = x_i + \delta(x_j - x_i) \) and \( f_j = f(x_j) \) for \( j = 0, 1, 2, ..., n \) with \( j \neq i \) are calculated.

**Termination tests**

The implementation of the Nelder-Mead method must include a test that ensures the termination process in a finite period of time. The termination test refers to the convergence or termination test performed when the simplex is relatively small, the function values convergence test is triggered when the \( f \) values are relatively close and the failure test is triggered when the values of the functions exceed a specific value.

The simplex method was used by Buclaghem and Letherman [23] to automatically optimize the passive performance of buildings. The optimization criterion was the objective function of minimizing comfort that combines the desired comfort temperature and the internal temperature of the building.

Al Hamoud (1997) optimized energy consumption using the simplex method and non-linear programming. The variables used are associated with the construction properties as well as the shape and orientation.

The simple algorithm of Nelder and Mead is a derivative optimization algorithm and constructs a simplex \( n \) dimension in the space of the independent variables. The cost function is estimated for each of the \( (n + 1) \) variables in the simplex. At each iteration step the simplex vector with the highest cost is replaced by a new vector which is achieved either by reflecting the largest cost function vector in the centroid of simplex or by expanding the simplex [24].

At the following example, we will use the Nelder-Mead algorithm, to find the minimum of the function \( f(x, y) = x^2 - 4x + y^2 - y - xy \), starting with the following vertices:

\[
V_1 = (0, 0), \quad V_2 = (1.2, 0.0), \quad V_3 = (0.0, 0.8)
\]

Given the vertices above, we have the following values for the function \( f \):

\[
f(V_1) = 0, \quad f(V_2) = -3.36, \quad f(V_3) = -0.16
\]

We set

- \( B = V_2 \) (gives the Best f-value)
- \( G = V_3 \) (gives a Good f-value)
- \( W = V_1 \) (gives the Worst f-value)
We will replace the vertex W=(0,0), and we set the points \( M = (B + G)/2 = (0.6, 0.4) \) and \( R = 2M - W = (1.2, 0.8) \).

Since \( f(R) = -4.48 < f(G) = -0.16 \), and \( f(R) \leq f(B) \), we should construct the E vertex, which is \( E = 2R - M = 2(1.2, 0.8) - (0.6, 0.4) = (1.8, 1.2) \).

Since \( f(E) = -5.88 < f(B) = -3.36 \), the new triangle has vertices \( V_1 = (1.8, 1.2), V_2 = (1.2, 0.0), V_3 = (0.0, 0.8) \).

A sequence of triangles which converges at the solution point (3,2) is generated from the continuation of the process, and gives the value \( f(3,2) = -7 \).

3. **General optimization Algorithms**

General Optimization is intended to find the general best solution for a problem when multiple local best solutions are known. Generic algorithms are considered to provide a better solution than local algorithms. There are several kinds of generic optimization algorithms that were developed depending on the application in which they were applied. The current work presents the DIRECT algorithm which is a combination of general and local optimization.

3.1. **The DIRECT algorithm**

The DIRECT algorithm is used in engineering applications for which there are no convincing assumptions and there are multiple minima and black box designs.

If an objective function \( f \) and the design space are considered \( D = D_0 \) the steps of the DIRECT algorithm are [25]:

**Step 1**: Design space \( D \) is normalized to be a unit super cube. Determines the center of \( c_i \) and estimates \( f(c_i) \). The algorithm starts from \( f_{\text{min}} = f(c_i) \) sets the estimation measure \( m = 1 \) and the repeat counter \( t = 0 \).

**Step 2**: Set the \( S \) group of the potential best boxes.

**Step 3**: Select each box \( j \in S \).

**Step 4**: Box \( j \) is divided in the following way:
1. Set Group \( I \) of dimensions with maximum length.Set \( \delta \) equal to 1/3 of the maximum side.
2. Select the function in points \( c \pm \delta e_i \) for all \( i \in I \) where \( c \) is the center of the box and \( e_i \) is the \( i \) unit vector.
3. Divide the \( j \) box containing \( c \) into thirds along the dimensions in \( I \) starting with the dimension with the smallest value \( w_i = \min(f(c + \delta e_i), f(c - \delta e_i)) \) and replaced in the dimension with bigger \( w_i \). Then, \( f_{\text{min}} \) and \( m \) are revised.

**Step 5**: Set \( S = S - \{j\} \) If \( S \neq 0 \) then repeat the process from step 3.

**Step 6**: Putting \( t = t + 1 \) if the repeat threshold or the limit of estimation has been reached, the algorithm stops. Otherwise the process goes to step 2.

The choice of boxes \( j \) is potentially optimal if:

\[
F(c_j) - K d_j \leq F(c_j) - K d_j
\]

Diakaki et al [26] using the DIRECT algorithm minimized the cost of life cycle and environmental impact using wall and window insulation as variables and the simulation was performed with a steady-state treatment model. In this work, the use of target programming and the global criterion method was found to be ideal for simple problems but scaling is a problem for this method.

Below, we give an example of DIRECT that has been used on the GP (Goldstein-Price) test function:

\[
f(x) = \left[ 1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \right] \\
\left[ 30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2) \right]
\]

where \( x_i \in [-2,2] \) for \( i = 1,2 \). The global minimum value of GP function is \( f_{\text{min}}=3 \). The figure below shows the plot of the function: 

13
3.2. Genetic algorithms

Genetic algorithms have proven their performance in various energy optimization problems such as optimizing building heating design [27]. These algorithms are a class of algorithms which begin by selecting a randomly produced population of solutions for the problem under consideration. They move from one generation of solutions to another, evolving new solutions using objective entities of assessment, selection, intersection and mutation. Generally, genetic algorithms function with solutions represented by a code instead of a set of initial variables. Typically, it is a solution represented by a series of bits. Each bit position is called a gene and the values acquired by each gene are called alleles. A basic genetic algorithm has three main operators performed at each iteration. The vectors are reproduction, intersection and mutation. At the breeding stage, the chromosomes or solutions of the existing breeding are copied to the next one with the same probability based on the value achieved for the objective function. At the intersection stage, randomly selected chromosome pairs are to be introduced into the next generation while the mutation stage includes an occasional random alternation of the allelic genes [28].

The selection factor for replication is useful for creating a new generation, improved in comparison with the previous one, while the intersection stage separates the population which handles the generations of the generated chromosomes and the mutation stage introduces the necessary data for the investigation area. Znuda et al 2007 developed a genetic algorithm aiming at a method for achieving a group of optimal architectural configurations based on the Goldberg algorithm. That is, they chose a coding strategy by starting the algorithm by random reproduction of the first population, and then each iteration produces a population better than the previous one based on the objective function being calculated. The algorithm terminates when the best solution is repeated in several iterations.

In another paper [29], they used the following mathematical procedure to describe the optimization problem:

\[
\text{Minimization } F(x_1, x_2, \ldots, x_n) \\
G(x_1, x_2, \ldots, x_n) \geq 0 \\
x_i \leq S_i
\]

There is a large number of objective \( F \) functions which should be minimized by convention and there are a number of \( G \) constraints which are required to be greater than or equal to zero. Moreover, each variable \( x_i \) is structured with specific \( S_i \) variables defined either as discrete variables or from boundary values. Objective functions and constraints must be alternating depending on how the problem is formed.
Genetic algorithms have proven to be much more efficient when optimization parameters are more than 10. The literature mentions [30] the minimization of energy by a genetic algorithm from the change of manufacturing options for the air conditioning of a building in tropical climate using the method of acceptance.

The minimization of costs and energy has been studied with genetic algorithms [31] using heat recovery and constructions as variables, while comparison of multiple and single objective functions has shown the advantage of objective functions despite the fact that this method has computational cost.

Genetic algorithms and multivariate analysis as well as artificial neural networks can be combined to evaluate performance and optimize composite production units based on structural indicators [32]. This combination initially determines the inputs and outputs for oriented neural networks and the genetic algorithm is based on multivariate analysis variables and collects the input data describing the input-output relationships for the modules. The linearity of the input data is then tested to assess the performance of the modules, normalize the input data based on the relationship determined, and divide into two subgroups in order to develop the neural networks. Different neural networks are then used to estimate the relationships between the input and output data, and a sensitivity analysis is performed by altering the input subgroup elements to achieve the minimum value while testing the genetic algorithms according to the selected parameters, and finally calculating the error between the actual output and output of neural networks and genetic algorithms. This combination of algorithms can be used in all building blocks for which suppliers set different economic, technical and environmental criteria. In addition, this combination method allows the continuous monitoring and improvement of composite units based on the overall productivity and reliability of the machine.

4. Conclusions

There are many different methods of using computers to optimize engineering designs. Below are brief details of some common optimization algorithms used for sustainable building optimization. All are heuristic methods, which means that they present no guarantee as to arriving at the true optimum solution but offer an efficient method with a high probability of approaching the optimum. "Direct search" covers methods that compare test solutions to the best ones calculated so far, with a strategy based on the results presented so far in order to determine the next test. Examples used in building optimization include:

- Search on a template, e.g. Hooke and Jeeves: Every dimension in turn is being tested. When no further improvement is possible, step size is reduced by half.
- Non-linear programming: a series of extensions allowing non-linear targets and constraints.
- Linear programming, including the simplex / Nelder and Mead method [33]: if objective operation and all constraints are linear, the optimal must fall to an extreme point.

The commonly used algorithms in building energy efficient design optimization can be grouped into three categories, namely evolutionary algorithms, derivative-free search algorithms, and hybrid algorithms. Note that hybrid algorithms are not new algorithms. They are combinations of different algorithms, often evolutionary algorithms with derivative-free algorithms. A statistical analysis of the core literature shows that evolutionary algorithms are the most commonly used category of optimization algorithms, accounting for approximately 60%. The rest are derivative-free search algorithms such as Hooke-Jeeves direct search algorithm [34] and hybrid algorithms. In the category of evolutionary algorithms, genetic algorithm (GA) [35] or its variations such as non-dominant sorting genetic algorithm (NSGA) [36] are dominant. A general trend to shift from normal GA to NSGA is noted, probably because NSGA is more suitable for solving multi-objective optimization problems, which are common in architectural design. Particle swarm algorithm is another frequently used evolutionary algorithm [37]. Although most researchers in the literature apply algorithms to building energy efficient design optimization without carefully studying the algorithm itself, several studies shed light on how effective and efficient different algorithms are in finding the optimal design solution. Wetter and Wright compared the Hooke-Jeeves direct search algorithm with a genetic algorithm and concluded that the latter performs better than the previous one and that the Hooke-Jeeves direct search algorithm can be trapped locally at optimum [38].Wetter and Wright compared the performance of eight algorithms: algorithm search algorithm, Hooke-Jeeves algorithm, PSO (Particle Swarm Optimization) algorithm,
PSO grid search, PSO-HJ hybrid algorithm, simple GA, Simplex algorithm Nelder and Mead and Discrete Armijo gradient algorithm, for their performance in minimizing cost functions with different smoothness[39]. They found that the hybrid algorithm achieved the most significant cost reduction with a higher number of simulations, and that the simple GA consistently approached the best minimum. However, the performance of other algorithms was not stable. A recommendation has also been made with regard to avoiding the Simplex algorithm and the Discrete Armijo grading algorithm if EnergyPlus is used to evaluate the cost function.

Last but not least, it is worth highlighting that the energy simulation engine is a key component in the building energy efficient design optimization workflow. Its function is to calculate the energy consumption or other energy related parameters of the building being designed and optimized. Among the studies included in the basic literature, EnergyPlus [40] is by far the most common energy simulation engine, and is the representative of the category of detailed and dynamic energy simulation programs. Other power assimilation engines found in the basic literature in this category are: DOE-2[41], TRNSYS[42,43], ESP-r[44], and IDA ICE[45].

In conclusion, energy-efficient design optimization is a philosophy and technique which assists architects and other professionals in designing buildings with high energy efficiency and better overall performance. This section draws upon the evidence presented to conjecture about future trends and challenges in computational optimization of sustainable buildings. There is a clear increase in the popularity of optimization for sustainable building design, especially for multi-objective optimization.

Acknowledgments

References

1. Al-Homoud MS., Optimum thermal design of office buildings. International Journal of Energy Research 21 (1997): 941-957
2. Shi Y., Developmental swarm intelligence: developmental learning perspective of swarm intelligence algorithms. International Journal of Swarm Intelligence Research 5 (2014): 36–54
3. Souomela, J., Survey of local algorithms. ACM Computing Surveys 45 (2013): 24–40
4. Hien, W. N., Poh, L. K., & Feriadi, H., The use of performance-based simulation tools for building design and evaluation-a Singapore perspective. Building and Environment, 35(8) (2000): 709–736.
5. Glaser, D. C., Feng, O., Young, J., & Xiao, L., Towards an algebra for lighting simulation. Building and Environment, 39(8) (2004): 895–903.
6. Castro-Lacouture, D., Sefair, J. A., Flórez, L., & Medaglia, A. L., Optimization model for the selection of materials using a LEED-based green building rating system in Colombia. Building and Environment, 44(6) (2009): 1162–1170.
7. Baglivo, C., Congedo, P. M., Fazio, A., & Laforgia, D., Multi-objective optimization analysis for high efficiency external walls of zero energy buildings (ZEB) in the Mediterranean climate. Energy and Buildings, 84 (2014): 483–492.
8. Yu, W., Li, B., Jia, H., Zhang, M., & Wang, D., Application of multi-objective genetic algorithm to optimize energy efficiency and thermal comfort in building design. Energy and Buildings, 88 (2015): 135–143.
9. Touloupaki, E., & Theodosiou, T., *Optimization of building form to minimize energy consumption through parametric modelling*. Procedia Environmental Sciences, 38 (2017): 509–514.

10. Bamdad, K., Cholette, M. E., Guan, L., & Bell, J., *Building energy optimisation under uncertainty using ACOMV algorithm*. Energy and Buildings, 167 (2018): 322–333.

11. Si, B., Tian, Z., Jin, X., Zhou, X., & Shi, X., *Inefficetiveness of optimization algorithms in building energy optimization and possible causes*. Renewable Energy, 134(2019): 1295–1306.

12. Kheiri, F., *A review on optimization methods applied in energy-efficient building geometry and envelope design*. Renewable and Sustainable Energy Reviews, 92 (2018): 897–920.

13. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*. Third Edition MIT Press (2009)

14. Ali Yalcin, Autar Kaw, *Chapter 09.01 Golden Section Search Method*. Holistic Numerical Methods(2012), Website: [http://mathforcollege.com/nm/mws/gen/09opt/mws_gen_opt_txt_goldensearch.pdf](http://mathforcollege.com/nm/mws/gen/09opt/mws_gen_opt_txt_goldensearch.pdf).

15. Patel A., Kumar V., Kumar Y., *Golden Section Search Optimization Technique for Maximum Power Point Tracking*. International Journal of Engineering Research and Applications 3 (2013): 1204-1209

16. R. Gayathri, G.A. Ezhilarasi, *Golden section search based maximum power point tracking strategy for a dual output DC-DC converter*, Ain Shams Engineering Journal (2017)

17. Fibonacci Search Technique. Website: [https://ipfs.io/ipfs/QmXoypizjW3WknFijNKLwHCnL72vedxijQkDDP1mXWo6uco/wiki/Fibonacci_search_technique.html](https://ipfs.io/ipfs/QmXoypizjW3WknFijNKLwHCnL72vedxijQkDDP1mXWo6uco/wiki/Fibonacci_search_technique.html).

18. Wenye Lin, Zhenjun Ma, *Using Taguchi-Fibonacci search method to optimize phase change materials enhanced buildings with integrated solar photovoltaic thermal collectors*. Energy 106 (2016): 23-37

19. Deyan Pang, Shouqiang Du, Jingjie JuPang, *The smoothing Fletcher-Reeves conjugate gradient method for solving finite minimax problems*. Science Asia 42 (2016): 40-45

20. Informatic 2013. *Optimization without Gradients: Powell’s method*, Website: [http://informatik.unibas.ch/fileadmin/Lectures/HS2013/CS253/PowellAndDP1.pdf](http://informatik.unibas.ch/fileadmin/Lectures/HS2013/CS253/PowellAndDP1.pdf).

21. Al Baali, M., Descent properties and global convergence of the Fletcher-Reeves method with Inexact Line Search, IMA Journal of Numerical Analysis 5 (1985): 121–124.

22. Saša Singer and John Nelder, *Nelder-Mead algorithm*. Scholarpedia (2009), Website: [http://www.scholarpedia.org/article/Nelder-Mead_algorithm](http://www.scholarpedia.org/article/Nelder-Mead_algorithm).

23. Buglaghem N.M., Letherman, K.M., *Numerical Optimization Applied to the Thermal Design of Buildings*. Building and Environment 25 (1990): 117-124.

24. Wetter, M., Wright, J., *A comparison of deterministic and probabilistic optimization algorithms for non-smooth simulation-based optimization*. Building and environment 39 (2004): 989-999.

25. Layne T. Watson, Jian He, *The Deterministic Global Optimization Algorithm DIRECT*. HPCS/BIS Tutorial (2008)

26. C. Diakaki, E. Grigorioudis, D. Kolokotsa, *Towards a multi-objective optimization approach for improving energy efficiency buildings*. Energy and Buildings 409 (2008): 1747–54.

27. Wright J.A., Loosemore H.A., Farmani R., *Optimization of building thermal design and control by multi-criterion genetic algorithm*. Energy and Buildings 34 (2002): 959–972.

28. Znuda E., Ghrab Morcos, N., Hadj Alouane, A., *Optimization of Mediterranean building design using genetic algorithms*. Energy and Building 39 (2007): 148-153.

29. Wang W., Rivard H., Zmeureanu R., *An object-oriented framework for simulation-based green building design optimization with genetic algorithms*. Advanced Engineering Informatics 19 (2005): 5-23.

30. Sahu M., Bhattacharjee B., Kaushik S., *Thermal design of air-conditioned building for tropical climate using admittance method and genetic algorithm*. Energy and Buildings 53 (2012): 1-6.

31. Palonen M., Hasan A., Siren K., *A genetic algorithm for optimization of building envelope and HVAC system parameters*. Eleventh International IBPSA Conference, In: Proceedings of the building simulation (2009).
32. Azadeh A., Mianaei, H.S., Asadzadeh S.M., Saberi, M., Sheikhalishashi, M., *A flexible ANN-GA-multivariate algorithm for assessment and optimization of machinery productivity in complex production units*. Journal of Manufacturing Systems 35 (2015): 46-75.

33. Nelder JA., Mead R., *A simplex method for function minimization*. Computer Journal 7 4 (1965): 308–13.

34. Holst J., *Using whole building simulation models and optimizing procedures to optimize building envelope design with respect energy construction and indoor environment*. Eighth International IBPSA Conference, In: Proceedings of the building simulation (2003).

35. Wright J., Alajmi A., *The robustness of genetic algorithms in solving unconstrained building optimization problems*. Ninth International IBPSA Conference, In: Proceedings of the building simulation (2005).

36. Brownlee I., Wright A., Mourshed MM., *A multi-objective window optimization problem*. In: Proceedings of the 13th annual conference companion on Genetic and evolutionary computation (2011).

37. Rapone G., Saro O., *Optimization of curtain wall facades for office buildings by means of PSO algorithm*. Energy Buildings 45 (2012): 189–96.

38. Wetter M., Wright J., *A comparison of deterministic and probabilistic optimization algorithms for non-smooth simulation-based optimization*. Building and Environment 39 (2004): 989 – 999.

39. Wetter M., Wright J., *Comparison of a generalized pattern search and a genetic algorithm optimization method*. Eighth International IBPSA Conference, In: Proceedings of the building simulation (2003).

40. Zhou G., Ihm P., Krarti M., Liu S., Henze G., *Integration of an internal optimization module within EnergyPlus*. Eighth International IBPSA Conference, In: Proceedings of the building simulation (2003).

41. Bichiou Y., Krarti M., *Optimization of envelope and HVAC systems selection for residential buildings*. Energy and Buildings 43 (2011): 3373–3382.

42. Carreras J., Boer D., Guillén-Gosálbez G., Cabeza L., Medrano M., Jiménez L., *Multi-objective optimization of thermal modelled cubicles considering the total cost and life cycle environmental impact*. Energy and Buildings 88 (2015): 335–346.

43. Xu J., Kim J., Hong H., Koo J., *A systematic approach for energy efficient building design factors optimization*. Energy and Buildings 89 (2015): 87–96.

44. Marco Manzan, Alberto Clarich, *Energy and daylighting optimization for an office with fixed and moveable shading devices*. Building and Environment 113 (2017): 175-184.

45. Hamdy M, Palonen M, Hasan A., *Implementation of pareto-archive NSGA-II algorithms to a nearly-zero-energy building optimization problem*. In: Proceedings of the building simulation and optimization conference (2012).