Comparison Between Five Stochastic Global Search Algorithms for Optimizing Thermoelectric Generator Designs

Mathieu Allyson-Cyr[a] and Louis Gosselin[a]*

[a]*Department of Mechanical Engineering, Université Laval, Québec, Canada*

*Corresponding author: louis.gosselin@gmc.ulaval.ca; Tel.: +1-418-656-7829

Article accepté pour publication dans : Numerical Heat Transfer, Part A: Applications, Volume 76, 2019

Comparison Between Five Stochastic Global Search Algorithms for Optimizing Thermoelectric Generator Designs

In this work, the best settings of five heuristics are determined for solving a mixed-integer non-linear multi-objective optimization problem. The algorithms treated in the paper are: ant colony optimization, genetic algorithm, particle swarm optimization, differential evolution and teaching-learning basic algorithm. The optimization problem consists in optimizing the design of a thermoelectric device, based on a model available in literature. Results showed that the inner settings can have different effects on the algorithm performance criteria depending on the algorithm. A formulation based on the weighted sum method is introduced for solving the multi-objective optimization problem with optimal settings. It was found that the five heuristic algorithms have comparable performances. Differential evolution generated the highest number of non-dominated solutions in comparison with the other algorithms.

Nomenclature

a_1, a_2, a_3	Divisions of the solution space
A_{total}	Heat collector surface of exchange [m^2]
c	Cell index, Eq. (5)
c_1	Personal learning coefficient (PSO)
c_2	Global learning coefficient (PSO)
C	Averaged Euclidian distance from Pareto front, Eq. (3)
C_n	Maximal value of C obtained by the worst algorithm
CR	Crossover rate (GA)
d_i	Smallest normalized Euclidian distance of point i, Eq. (2)
D_{ijk}	Density function
$f_k(i)$	Function values of the objective k of point i
$f_k^{\min}$, $f_k^{\max}$	Minimal and maximal function values of the objective k

F	Objective function
$\lvert F \rvert$	Total number of points on the Pareto front for a chosen algorithm
F'''	Three dimensional feasible domain
G_{best}	Best score of the swarm (PSO)
H	Entropy
H_n	Normalized entropy
Ite	Number of iterations to reach 98% of the maximal objective function
k	Number of ants in a colony (ACO_R)
N	Total number of solutions, Eq. (5)
n_{total}	Number of thermoelectric modules
pCR	Crossover probability (DE)
Pop	Population
P_{best}	Best personal score (PSO)
$P*$	Pareto front
P_{net}	Net electric power output [W]
q	Intensification factor (ACO_R)
r	Euclidian distance between the solution and a specified cell
s	Solution index, Eq. (5)
s_1, s_2, s_3	Weight ratio coefficients of each objective
$w_{inertia}$	Inertia weight (PSO)
w_{damp}	Inertia weight damping (PSO)

Greek symbols

β	Scaling factor (DE)

ζ	Pheromone evaporation rate (ACO_R)
ρ_{ijk}	Normalized density function
σ	Standard deviation
Ω	Influence function
$\Omega_s\left(r_{s\to c}\right)$	Influence function between solution s and cell c, Eq. (5)

Acronyms

ACO	Ant colony optimization
ACO_{MV}	Ant colony optimization (mixed-variable problem)
ACO_R	Ant colony optimization (continuous domain)
DE	Differential evolution
EA	Evolutionary algorithm
PDF	Probability density function
GA	Genetic algorithm
PSO	Particle swarm optimization
RSD	Relative standard deviation
SI	Swarm intelligence
TEG	Thermoelectric generator
TLBO	Teaching-learning basic optimization

1. Introduction

Modern engineering design problems are complex problems that can become quite large. They typically involve discrete and continuous variables, as well as non-differentiable and non-linear functions. As a result, these optimization problems are often difficult to solve, in particular with “traditional” optimization techniques.

Stochastic global search algorithms are a class of derivative-free optimization methods that are an appealing option for solving these problems [1, 2]. This family of algorithms uses stochastic and direct-search methods to find good approximate solutions to complex problems with little to no prior knowledge of the optimization problem. These algorithms are often referred to as heuristics and can be classed into evolutionary algorithms (EAs), such as genetic algorithm (GA) and differential evolution (DE), or swarm intelligence (SI), like particle swarm optimization (PSO). Due to their inherent stochastic features, these algorithms can lead to different solutions to the same problem each time they are used and thus the optimality is not guaranteed. Nonetheless, these algorithms can converge faster than other methods and can find nearly optimal solutions efficiently when used correctly [3].

The convergence mechanism of heuristic algorithms can be adapted through two main forces during the search for the global minima: diversification and intensification. Diversification is a term used to describe the exploration of the overall design space, whereas the intensification characterizes the local search of a specific portion of the design space. Both forces are contradictory and complementary, and must be balanced to reach nearly optimal solutions efficiently [4]. For the vast majority of existing heuristics, algorithm settings must be tuned prior to the optimization [5–7]. These parameters directly influence the convergence mechanisms and can be problem dependant. Setting the algorithm parameters for complex nonlinear mixed-variable problems becomes a crucial part of the optimization process for most of the heuristic algorithms. Various methods exist to determine optimal settings. Most of these methods are considered as an offline tuning which consists in determining the best algorithm parameters before actually using the algorithm for solving a specific optimization

problem [8]. Alternatives also transform the offline tuning problems into a continuous optimization problem to be used with continuous optimization techniques [9].
With appropriate settings, heuristics can solve various types of problems. The algorithms presented in this paper were initially developed for solving continuous single-objective problems. Nonetheless, these algorithms can also be used to solve multi-objective optimization problems. According to the literature, one of the most common methods is the weighted summation approach which has been intensively studied [10]. This approach is a general scalarization method combining all the objective functions into a single function, where the objectives are summed with weighted parameters. Under some limitations, this approach can be used with single-objective heuristics to solve mixed-integer, non-linear, multi-objective optimization problems. However, the weighted sum method has some drawbacks that can lead to inefficient Pareto front representation [11]. Few performance comparisons have been found in the literature between different stochastic global search algorithms with this method.

When confronted to a complex optimization problem, the selection of a proper heuristic algorithm in terms of optimality of the solutions and acceptability of the computational time is not straightforward, and comparison algorithms can be quite tedious [12]. In addition, the settings suggested for various heuristics in theory or abstract models are not necessarily representative of real-life models [8]. There is thus a need to develop more knowledge on the selection of an appropriate heuristic algorithm for complex real-life optimization problems with multiple local minima objectives and a high number of variables.

In this paper, five different population-based stochastic heuristic algorithms are compared for the optimization of the design of a thermoelectric device, which is a

mixed-integer nonlinear constrained problem (Ref. [14]). First, the impact of the algorithm internal parameter settings on the convergence speed, repeatability, and capacity to reach optimal solutions is investigated. Then, a multi-objective optimization problem with a weighted sum method is solved using the best setting identified in the previous section. Convergence and diversity metrics are used to compare the performance of the population-based heuristics.

2. Heuristic algorithms

This section presents the five selected population-based heuristic algorithms to be compared. The problem solved in this study is described below, in Section 3, and consists in optimizing a thermoelectric generator design. This reveals to be a multi-objective nonlinear optimization problem with both continuous and discrete variables. The main features of each algorithm are summarized in the sub-sections below. References are provided for readers interested to learn more about the details of each heuristic. The algorithms have been implemented in *Matlab*.

It should be noted that the algorithm presented here were originally created for continuous optimization problems. Since the test case is a mixed-variable optimization problem, the discrete variables must undergo a specific treatment. Since all the discrete variables of the thermoelectric problem are categorical variables, there is no intrinsic ordering. Therefore, the integer variables are considered as continuous variables and then rounded to their nearest valid indices before evaluating the objective function.

2.1 Genetic algorithm (GA)

Genetic algorithms are one of the most popular EAs that use the concept of natural selection to create offspring. A first version of the algorithm has been introduced in Ref.

[15]. The bio-inspired concepts of selection, crossover, and mutation are used to produce better offspring for the next generation (i.e. better solutions).

According to literature, GAs have been highly studied and applied to many fields such as heat transfer problems [16], HVAC systems [17], porous medium combustion [18], flow-shop scheduling [5], and thermoelectricity [19]. The algorithm has also been adapted to tackle mixed-discrete optimization problems [20, 21] and multi-objective optimization [22].

The selection is the first step and it consists in selecting promising parents for the generation of offspring. Several methods can be used to select the individuals in the population. In this paper, three selection methods are retained: uniform selection, roulette selection, and tournament selection. The first selects a parent with a normal distribution based on the value of its solution and the number of parents. The roulette selection method simulates a roulette wheel in which every individual is present with the area proportional to its solution value. The tournament method chooses a group of 4 individuals randomly in the population and then selects the best ones to become a parent.

A crossover is then performed between two parents to generate new offspring. The crossover ratio (CR) is used to determine the number of individuals in the population that will become the parents (and the number of offspring). The method used to make a crossover child can also be adapted. In this paper, three crossover methods are retained: scattered, single point, and double points. Each of these methods provide a different way to determine from which parent the child will inherit each design variable. The scattered method creates a random binary vector equal to the number of variables and then gives one of the parent variable to the child depending on the value of the binary vector. The single point method selects randomly a value between 1 and the

maximal number of variables. The child receives all the variable values of the first parent before this point and the rest comes from the second parent. The double point method is similar but two points are randomly selected. The first parent gives the variable values between the two points to the child and the rest is passed from the second parent

The last step is the mutation. The mutation replaces randomly some variable values in some individuals of the population. With the GA used here (from the *Matlab* OptimizationToolbox, Release 2016b), the number of individuals experiencing mutations is fixed to the remaining fraction of individuals that has not been subject to the crossover $(1-CR)$. An adaptable feasible method is used here for the mutation in order to satisfy variable bounds. The pseudocode of the algorithm can be summarized as:

Step 1. State variable bounds, algorithm parameters and termination criterion.

Step 2. Initialize the population, the crossover fraction CR and the selection and crossover methods.

Step 3. Select individuals to become parents.

Step 4. Perform the crossover on the parents to generate Pop×CR children.

Step 5. Perform the mutation on Pop×(1–CR) individuals.

Step 6. Re-form a population combining mutants, children and initial individuals and then eliminate the worst individuals.

Step 7. Repeat steps 3 to 6 until the stopping criterion is met.

2.2 Particle Swarm Optimization (PSO)

PSO is a bio-inspired heuristic mimicking the behavior of bird flocks or fish schooling [23]. The algorithm has been developed for solving continuous nonlinear problems. Nonetheless, the algorithm can be adapted to solve mixed problems by converting

discrete variables into continuous values as explained in Ref. [24]. PSOs have been successfully used in many real-world applications like heat exchangers [25, 26], thermoelectricity [27, 28], inverse heat transfer problems [29, 30], mechanical designs [31], and geothermal power plants [32]. Improvements to the algorithm have been developed for multi-objective optimization [33], easier parameter tuning [34], better search efficiency [35], or to handle local minima more efficiently [36].

The algorithm explores the design space, searching for the optimal solution by changing the trajectories of each "particle" of the swarm. Each particle is a member of the population and a solution vector of the optimization problem. Each individual particle moves in the multidimensional space toward the optimal solution by modifying its position using the information of its best personal score (P_{Best}) and the best score of the entire swarm (G_{Best}).

The velocity of each particle is modified using three elements that can be adapted with different parameters. The first element is the velocity of the previous iteration. The parameter $w_{inertia}$ is the inertia weight affected to the velocity to control the exploration factor of each particle and minimize the risk of the algorithm getting trapped into a local minimum. The second and third elements use the information of P_{Best} and G_{Best} to change the direction of the particle. The personal learning coefficient c_1 and the global learning coefficient c_2 are used to control the intensification of the search. In addition, w can be damped at each iteration with the parameter w_{damp} to gradually reduce the inertia weight $w_{inertia}$ and increase the intensification of the search when the swarm is close to the global minima. The last parameter is the velocity limitation. Without control over the minimal and maximal value of the velocity, the

swarm could “explode” with very high velocity, moving the particles outside the limits of the design space. The pseudocode of the algorithm can be summarized as:

Step 1. State variables bounds, velocity limits, algorithm parameters and termination criterion.

Step 2. Initialize the population and the algorithm parameters.

Step 3. Evaluate the objective function of each particle. P_{Best} and G_{Best} can be changed if they are better than their actual values.

Step 4. Modify each particle position. $w_{inertia}$ can be damped at this step if a damping parameter is defined.

Step 5. Repeat steps 3 and 4 until the stop criterion is met.

2.3 Ant Colony Optimization for continuous variables (ACO_R)

ACO is an algorithm inspired from the behavior of ants foraging. The algorithm was initially developed to solve discrete problems [37, 38]. One particularity of the algorithm is how it recreates the movement of ants from the nest to the foraging area and vice-versa following the shortest path. The ants are solutions that follow one of the paths that are potential solutions in the solution space. When an ant follows a specific path, it has the solution component from that particular path. When an ant moves between the food source and the nest, it releases a pheromone trail that other ants can smell to change their path. The pheromone trail is used to probabilistically sample the search space (paths). The shorter the path, the stronger the concentration of the pheromone trail, increasing the probability that ants move to that specific path. This bio-inspired mechanism allows ants to interact with other nest mates to move through shorter path in the design space (minimal cost). An evaporation parameter is also included to the pheromone trail in order to control the premature convergence to a path

that might not be the global minimum. At each iteration, the concentration of the pheromone trail is lowered to reduce the attraction of the other ants and increase the exploration of the design space by other ants.

ACOs have been widely used in many fields of application. To name a few, ACOs has been successfully applied to heat transfer problems [39, 40], thermoelectricity applications [41] and water distribution systems [6]. ACO has also been combined with other heuristic algorithms such as PSO to solve inverse heat transfer problems [42, 43] for higher effectiveness and overall robustness. Variations of the algorithm exist to solve other kinds of problems. ACO_R is a variant for continuous domain [44] and ACO_{MV} is a variant for mixed-variable problems [45]. For the mixed-variable problem of the present test case, ACO_R was used with a treatment of the discrete variables similar to that in $ACO_{MV,}$ where discrete variables are treated as categorical variables, similarly to Ref. [46].

The main idea behind ACO is the pheromone trail used for incremental construction of solutions. A finite set of possible paths exist and a probability is given from best to worst to each of them depending on the pheromone value. However, with continuous variables, the concept of pheromone is different. Instead of a discrete probability distribution, a weight is given to each solution in the population rated from best to worst with a probability density function (*PDF*). The weight determines the attractiveness of a solution during the construction process and a probability for an ant to choose this solution is set based on the weight value. The algorithm parameter q is the intensification factor used in evaluating the weight. A small value of q increases the weight for the best-ranked solutions while a high value of q makes the weight more uniform among the ranked-solutions. Then, the probability to select the solution is calculated for each solution based on the weight and their rank.

In the first step, k ants (potential solutions) are randomly generated. In a second step, the algorithm starts the construction of new solutions variable by variable by generating Gaussian random variables. The number of new solutions created can be adapted. In this paper, the sample used to construct new solutions is fixed to half the population. For each variable of each solution, a standard deviation is calculated including the deviation-distance ratio ζ that works in a way similar to the pheromone evaporation rate in ACO. ζ affects the long term memory by reducing the search for already explored points. Therefore, the convergence rate is often lower with a higher value of ζ. Variable by variable, the algorithm select a solution with a roulette wheel selection scaled with the probability. Then, a Gaussian random variable is generated from the standard deviation and the value of the variable from the selected solution. The pseudocode can be summarized by the following steps:

Step 1. State variable bounds, algorithm parameters and termination criterion.

Step 2. Initialize the population.

Step 3. Calculate the Gaussian functions w and the probability associate to each solution rank.

Step 4. Construct solutions (new solutions). For each individual variable, select the Gaussian kernel function with the probability to select each Gaussian function w. Then, generate random Gaussian variable.

Step 5. Update the population by eliminating the worst solutions.

Step 6. Repeat steps 3 to 5 until the stop criterion is met.

2.4 Teaching-Learning Basic Algorithm (TLBO)

TLBO is inspired by the apprenticeship of a teacher with learners by exchanging information to reach global minima [47–49]. Similarly to other heuristic algorithms, TLBO is a population-based algorithm where each individual of the population is a

learner and the teacher is the best individual in the population. However, one particularity of this algorithm is that it has no internal parameters to fix, which makes it easy to adapt to any optimization problem. However, since the algorithm is developed to solve non-constrained nonlinear continuous problems, its capability to solve mixed-integer non-linear problems is not clear.

TLBOs have been used to solve many kind of optimization problems, such as heat exchanger optimizations [50], thermoelectric cooler [51], and heat pipes [52]. The algorithm can be used to solve multi-objective optimization problems [53].

The algorithm has two distinct phases. The first phase is the teaching phase where the best learner (i.e., the selected teacher) tries to provide knowledge to the other learners to improve the overall mean score of the class. In the second phase, the learners interact with each other to improve their best score. The best learner among the two teaches his knowledge to the other one. This is done by adding to the worst learner a fraction of the difference in the solution vector between the two learners. This fraction is randomly generated for each interaction between two learners [47]. The intensity of this exchange could be increased, but here, the ratio of exchange is chosen randomly for each variable of the solution. It should be noted that for both phases, the objective function is evaluated for the entire population. Therefore, the algorithm required twice the number of objective function evaluations as the algorithm presented in this paper. Finally, the best solution (learner) is updated to become the teacher for the next iteration. The pseudocode can be summarized as:

Step 1. State variable bounds, algorithm parameters and termination criterion.

Step 2. Initialize the population.

Step 3. Calculate the population mean and identifying the best solution to become the teacher.

Step 4. Teaching phase: modifying solutions with the best solution (i.e., the teacher). Update the best solution if one of the learners is better.

Step 5. Learning phase: a learner is compared to another one randomly where the better solution is used to modify the other learner. Update the best solution if one of the learners is better.

Step 6. Repeat steps 3 to 5 until the stop criterion is met.

2.5 Differential Evolution (DE)

DE is an algorithm with an approach similar to that of GAs (i.e., mutation, crossover, selection). The algorithm has been introduced by Ref. [54] to solve nonlinear continuous problems.

According to the literature review, DEs have been less studied than other heuristics. Nonetheless, authors have used DE in some applications such as optimal shell-and-tube heat exchangers [55], heat exchanger network synthesis [56], and other various engineering problems [57, 58]. Variations of DEs also exist to solve multi-objective optimization problems [59].

At each generation, the algorithm selects randomly a target vector and two other solution vectors from the population pool and performs a mutation using a scaling factor β. This factor affects how different the mutant vector will be from the target vector. Afterwards, a crossover is done between the target vector and the mutant vector to obtain the trial vector with a crossover probability pCR. The crossover is performed on each variable of the vector. Higher pCR increases the probability that the trial vector will have variable values from the mutant vector. The best solution between the trial vector and the target vector is the solution retained. The pseudocode can be summarized by:

Step 1. State variable bounds, algorithm parameters and termination criterion.

Step 2. Initialize the population.

Step 3. Perform the mutation with three randomly selected vectors.

Step 4. Perform the crossover between the mutant vector and the trial vector.

Step 5. Select the best solution between the trial vector and the target vector. Retain the best one in the population for the next generation.

Step 6. Repeat Steps 3 to 5 until the stop criterion is met.

3. Test case: Thermoelectric generator design optimization

As mentioned above, the five selected algorithms have been analysed on a series of continuous problems. However, few comparison exercises were found, and in particular, the capability of these algorithms to solve non-linear mixed variable problems is not well documented. The optimization problem described in the present section has been used for that purpose. It consists in optimizing the design of a thermoelectric generator system developed for heat recovery applications, see Ref. [14].

Thermoelectric generators (TEGs) are devices capable of directly converting heat into electricity by using the Seebeck effet. Featuring no mobile parts or working fluids, highly durable TEGs are attracting more and more attention for heat recovery applications. However, the high cost and low efficiency of actual thermoelectric materials is the main limitation for commercial use. The thermoelectric generator model used in this paper is based on the one developed in Ref. [14], and therefore is not repeated here. The heat source has constrained temperature and heat flux distributions over its surface. In order to achieve a significant temperature difference through the thermoelectric modules, a cooling system is installed on the cold side of the TEG. Fig. 1 shows a schematic view of the system. [Figure 1 near here]

3.1. Objective functions

The objective of this design optimization is to maximize the electric power output while minimizing the cost of system. The cost of the TEG system is related to the number of thermoelectric modules and the total surface of exchange of the cooling system. Thus, a total of three objectives should be simultaneously optimized. A weighted sum method is used here since most of the algorithms presented were primarily developed for single-objective problem. Therefore, the objective function is:

$$F = s_1\left(-P_{net}\right) + s_2\left(n_{total}\right) + s_3\left(A_{total}\right) \tag{1}$$

where the coefficients s_i are the weight ratio of each objective i, P_{net} is the net power obtained from the difference between the electric power output and the pumping power for the cooling system, n_{total} is the total number of modules of the TEG, and A_{total} is the total surface of exchange of the heat exchanger (cooling system). In order to obtain the equivalent of a Pareto front corresponding to all three objectives, multiple combinations of weight ratios are tested with each algorithm. Every combination of weight ratio varying from [0:0.2:1] is tested for a total of 216 solutions for each run. Afterwards, a non-dominated sorting is performed on the solution set to eliminate the dominated solutions. It should be noted that the weighted summation approach is unable to generate solution of non-convex portion of a Pareto optimal front [11]. [Figure 2 near here] [Table 1 near here]

3.2. Model description

In this paper, the fixed hot-side heat flux and temperature distribution shown in Fig. 2 was used to test the algorithms. The surface is divided into a 12 × 12 grid for a total of 144 cells of equal dimensions. The list of 165 variables (both discrete and continuous) of the model is presented in Table 1.

4. Analysis of the effect of heuristic algorithm parameters on performance

The “best” algorithm settings can be different from a problem to another and can affect the comparison exercise proposed here. Different techniques have been proposed to fine-tune the parameters of evolutionary algorithms (e.g., [60] and [61]), but for the type of problems addressed here, only scarce information or rules of thumb were found in literature to select the optimal algorithm parameters. Therefore, it was decided to investigate and document the effect of these parameters on the performance of the algorithms. A specific case was used for this purpose, with weight ratio values $\{s_1, s_2, s_3\}$ of $\{1, 0, 0\}$. In other words, this corresponds to the problem of maximizing the net power output only. Note that a preliminary analysis (not shown here) has verified that a change in the weights did not affect significantly the main findings of this section (i.e., which algorithm parameters work better). The stopping criterion is when the number of iterations reaches 100. The number of individuals in the population is set to 2000. The initial population is different for each run and is generated using a Latin hypercube sampling (LHS) rather than creating a uniformly distributed random population. It was demonstrated that the initial population generated from LHS is more uniform across the design space which leads to faster convergence speed and higher diversity of optimal solutions [62]. Table 2 presents the list of parameters tested and their values. One should note that every possible parameter combination of Table 1 has been tested. To verify the repeatability, a total of 5 runs are done with each algorithm for each parameter combination. [Table 2 near here]

4.1. Impact of algorithm parameters on convergence rate, repeatability and optimal solutions

The combination of algorithm parameters can either increase the exploration of the search space or increase the intensification of the search around local or global minima.

For a given problem, a trade-off must be made between different algorithm parameters to reach a global minimum in less iterations. However, the effect of those algorithm parameters on the performance (i.e. convergence rate, repeatability and achievement of global minima) is not always clear. For example, some algorithms can reach optimal performance with a wide range of algorithm parameters combinations making them easier to use, such as PSO [32]. One might thus be interested in determining the effect of those algorithm parameters on the algorithm performance. Therefore, this section features an analysis of the effect of algorithm parameters of GA, PSO, ACO_R and DE. One should remember that TLBO does not have internal parameters to choose.

In this section, the algorithm parameters from Table 2 are tested and compared for each algorithm. For every possible combination of algorithm parameters, the performance of an algorithm was assessed by calculating:

(1) The best solution obtained among the 5 optimization runs;
(2) The coefficient of variation or relative standard deviation (RSD), which is the ratio of the standard deviation over the mean for the 5 optimization runs. A low RSD value indicates a better repeatability;
(3) The number of iterations required to reach 98% of the maximal objective function (Ite). The average Ite among the five runs is calculated.

These metrics are reported in Figs. 3 to 5 as a function of the algorithm parameters.

In Fig. 3, one can see the impact of the crossover ratio CR and the selection method on the performance of the GA. The crossover method has been fixed to the scattered function. In Fig. 3a, a crossover ratio of 0 (i.e., mutation ratio of 1) provides poor solutions with either of the selection methods. In addition, high mutation reduces considerably the convergence speed and the reliability. It can be seen from Figs. 2b and

2c that the number of iterations required and the RSD tend to be higher with CR below 0.6. With CR over 0.6, each selection method is able to reach satisfying solutions with less iterations and with a better reliability. Nonetheless, for a better reliability, the tournament selection method should be used since the RSD is the lowest as presented in Fig 2b (~0.015). Therefore, proper parameters for the test case could be CR between 0.6 to 1.0 while using the tournament selection method and the scattered crossover method. [Figure 3 near here]

In Fig. 4, the effect of the personal and global learning coefficient (c_1 and c_2) of PSO on the performance is presented. The inertia weight $w_{inertia}$ and inertia damping w_{damp} have been fixed to 1 and 0.99 respectively based on Ref. [7]. One should note that, reducing w_{damp} resulted in premature convergence of the algorithm while removing the damping resulted in too much exploration leading to much lower convergence speed. In Fig. 4a, the results show that the value given to c_1 and c_2 has no significant impact on the solution obtained. Increasing the intensification of the search (i.e. by increasing c_1 and c_2) seems to have no effect on the performance. However, having c_2 = 0 is not an appropriate choice because the algorithm does not converge as shown in Fig. 4a. Furthermore, the impact of c_1 is negligible for this test case. From Fig. 4b, PSO seems to have an overall good reliability since the RSD is relatively low (~0.01). As for the convergence speed, the results from Fig. 4c show that higher values of c_1 and c_2 increase the number of iterations required to reach the final solution. With a higher weight given to personal and global learning, some individuals in the population are more easily attracted toward local minima. Hence, the local minima obstructs the swarm in the search of the global minima, resulting in lower convergence

speed. Therefore, with $w_{inertia}$ =1 and w_{damp} =0.99, appropriate c_1 and c_2 values could be between 0.4 and 1.4 for this test case. [Figure 4 near here]

In Fig. 5, the effect of the intensification factor q and the deviation-distance ratio ζ of ACO_R on the performance is presented. In Figs. 4a and 4c, the results show that a high value of ζ reduces the capacity of the algorithm to reach good solutions as well as the convergence speed. Consequently, lower ζ values (between 0.3 and 0.6) appear to be the best choice. In Fig. 5a, with ζ between 0.3 and 0.6, q can cover a wide range without affecting the capacity of the algorithm to reach good solutions. However, q has a direct impact on the convergence speed of the algorithm. In Fig. 5c, it can be seen that lower values of q (i.e., increasing the weight for high-ranked solutions) reduce the total number of iterations required for the algorithm to converge. Thus, good parameters combination could be ζ ~ 0.4 and q below 0.01 for fast convergence and good capacity to reach satisfying solutions. In Fig. 5b, RSD is globally higher with every parameter combinations and no tendency can be observed, which suggests that reliability provided by ACO_R is generally low for this test case. [Figure 5 near here]

In Fig. 6, the effect of the crossover probability pCR and scaling factor β of DE on the performance are presented. Fig. 6a shows that higher pCR leads to better solutions. In addition, having lower β also improves the solutions. Nonetheless, the effect of β is less present with high pCR. One should note that a pCR value of 1 makes all trial vectors mutant which prevents the algorithm from converging correctly. In Fig. 6b, the RSD is relatively low and constant, which reveals that the algorithm has a good reliability overall. The convergence speed in Fig. 6c shows that high pCR and low β reduce the number of iterations required to reach convergence. In the end,

proper parameters for DE for this test case are approximatively $pCR \sim 0.8$ and $\beta \sim 0.01$. [Figure 6 near here]

To better compare the different algorithms, the resulting net power output versus the RSD is reported in Fig. 7 for every combination of algorithm parameters tested for GA, PSO, ACO_R and DE. The power output versus the required number of iterations (average over the 5 runs with the same settings) is also shown in this figure. Each point in Fig. 7 represents a single combination of parameters.

In Fig. 7a, it can be seen that DE (in blue) has to lowest RSD values and setting different algorithm parameters mostly affect the best solution obtained. On the other hand, PSO (in green) has higher RSD values as well as higher P_{net} globally with most of the algorithm parameters tested. Nonetheless, the maximal net power achieved does not change significantly with the PSO settings. ACO_R (in red) appears as a "compromise" between PSO and DE. In Fig. 7a, it can be seen that ACO_R has most of its points (in red) clustered between DE (in blue) and PSO (in green) for the RSD value. In addition, most of the cluster of points is located close to higher power output values. Therefore, results from Fig. 7a suggest that ACO_R has both higher repeatability and better capacity to reach good solutions with multiple algorithm parameter combinations. In contrast, GA (in black) has a more spread out cluster and most of its points reach lower net power output as well as higher RSD.

In Fig. 7b, the net power output versus Ite is shown for each combination of algorithm parameters. It can be seen that DE (in blue) has the slowest convergence rate with most of its parameter combinations. In general, both GA and PSO required around 25 to 50 iterations to reach nearly optimal solutions. Similarly to their RSD, changing the algorithm parameters for these algorithms has a significant impact on the convergence speed but they still remain faster than DE. As for ACO_R, Fig. 7b reveals

that the choice of algorithm parameters has an important impact on the convergence speed. In fact, the required number of iterations for ACO_R varies from 5 to 95. In addition, it can be seen that the points with the lowest or the highest required number of iterations lead to less net power output. Therefore, the convergence rate and level of optimality are highly affected by the ACO_R setting. [Figure 7 near here]

4.2. Algorithm performance comparison with proper algorithm parameters

Based on the previous figures, a set of algorithm parameters has been chosen for each algorithm. These choices offered a satisfying trade-off between the capability to achieve nearly optimal solutions, as well as good reliability and convergence rate. The selected algorithm parameters, the maximal objective function value, and the RSD of each algorithm are reported in Table 3. The convergence of GA, PSO, ACO_R, DE and TLBO with the algorithm parameters from Table 3 is reported in Fig. 8. The solid lines are the average solution of all five runs at each iteration, while the gray areas are the range between the best and worst solutions among the five runs. The size of the gray area is directly related to the RSD. [Table 3 near here]

It can be seen in Fig. 8 that the convergence of both DE and TLBO is slower. Furthermore, DE even seems not to have fully converged even after 100 iterations. Nonetheless, DE has a smaller "gray area" revealing its low RSD value. Moreover, DE was able to achieve the best solution of all (i.e., a net power of 12,004 W). Even if TLBO required no algorithm parameters to set, it has to perform twice the number of objective function evaluations, making it more time-consuming. Globally, it can be concluded that for the present problem, DE is highly reliable and reaches good solutions but converges slowly. On the contrary, TLBO has slow convergence and lower capability to reach nearly optimal solutions. [Figure 8 near here]

The low rate of convergence of DE and TLBO can be compensated with GA, PSO and ACO_R. However, the gray area of PSO in Fig. 8 is important, meaning that different runs might be required and that it is likely that the solution of a run will not be the global minima. Regardless, a wide range of coefficients c_1 and c_2 can be selected to reach similar solutions as shown previously in Fig. 7. With a similar value of the maximized objective function P_{net} as that provided by PSO, GA required less iterations and had a better repeatability as presented in Fig. 8. Overall, the algorithm that reaches the best solution is DE, the algorithm that has the fastest convergence speed is ACO_R, the algorithm that as the best reliability is DE and TLBO. Other studies on DE compared to other population-based heuristics support this conclusion [7, 63, 64].

The conclusions on the performance of all five algorithms are summed up in Figure 9. Based on the results presented above, the algorithms have been ranked based on the different investigated criteria for the present optimization problem. The further an algorithm sits on a given axis, the better it is. The "robustness to parameter variation" is an added criterion that evaluates how sensitive an algorithm is to changes of its internal settings. This "qualitative" criterion is assessed from the objective function RSD from all the algorithm parameter combinations. Fig. 9 expresses that PSO is the only algorithm where changing the algorithm parameters has little to no impact on the solution reached. [Figure 9 near here]

5. Impact of heuristics on sets of non-dominated solutions

The multi-objective optimization problem introduced above is solved with the five heuristics compared here using the algorithm parameters found in Table 3. As mentioned before, the weighted sum method is used to build a single objective function combining the original three objectives, see Eq. (1). Optimizations are done for every

combination of weights s_i between [0:0.2:1] for a total of 216 optimization problems for each algorithm. The same number of individuals and maximal number of iterations as above is used in this section. Once again, five different runs are done for each algorithm for a total of 5400 optimization runs. This section compares the sets of solutions that are obtained as a function of the optimization algorithms.

5.1. Creation of Pareto front

With the total of 5400 solutions obtained, it is possible to create the equivalent of a Pareto front. A non-dominated sorting is performed on the 5400 points to keep only the non-dominated points (1077 points in this case) which were then reported in Fig. 10. This represents a good estimation of the Pareto front for this problem. In fact, Fig. 10 was compared to the multi-objective optimization solutions for the same thermoelectric model reported in Ref. [14]. Both sets of non-dominated points were found to have nearly identical hyperspace surfaces over the solution space. This demonstrates the validity of the weighted sum method used for this test case. [Figure 10 near here]

The Pareto front P* from Fig. 10 is obtained by selecting non-dominated points from the solutions provided by all the algorithms. It is interesting to find out which points from P* originate from which algorithm along with their location over the Pareto front. In Fig. 11, the solutions are represented on the Pareto front as a function of the algorithm from which they originate. In addition, the percentage of points on P* provided by each algorithm is reported in Table 4. These percentages were calculated to determine which algorithms were able to find the best solutions with the weighted summation approach. It was found that all heuristics were able to provide non-dominated solutions, with DE providing the most (32%) and PSO, the least (10%). [Figure 11 near here]

5.2. Metric for convergence

To evaluate the performance of each algorithm, an approach similar to the metric for convergence of Ref. [65] is used. The convergence metric typically evaluates the convergence at each generation of a given set of points to a reference set which is the allegedly "true" Pareto front (Fig. 10 or P* in this case). With the weighted sum method, each optimization (i.e., each point on the Pareto front) is independent and thus, a "classic" convergence analysis does not provide any useful information on the convergence speed. Instead, only the final front of each algorithm was compared to the true Pareto front.

The set of points from Fig. 10 forms a reference set P^*. For each point i of each run of each algorithm, the smallest normalized Euclidian distance d_i from P^* is calculated with:

$$d_i = \min_{j=1}^{|P^*|} \sqrt{\sum_{k=1}^{M} \left(\frac{f_k(i) - f_k(j)}{f_k^{\max} - f_k^{\min}} \right)^2} \tag{2}$$

where $f_k(i)$ are the function values of the objective k of point i from the algorithm studied, $f_k(j)$ are the function values of the objective k of point j from P^*, $f_k^{\min}$ and $f_k^{\max}$ are the minimal and maximal function values of objective k from the set P^*. The averaged distance is the retained value for each of the five optimizations for each algorithm:

$$C = \frac{\sum_{i=1}^{|F|} d_i}{|F|} \tag{3}$$

where $|F|$ is the total number of points on the front for the chosen algorithm. The lower the value of C, the closer the algorithm to the Pareto front of Fig. 10. Since 5 runs are performed for each algorithm, a total of 25 metric values C are calculated. Then, to

keep the metric values between 0 and 1, the metric values C are normalized by the maximal value obtained from the worst algorithm (C_n). Therefore, the worst algorithm has a score of 1 while the others are below.

5.3. Metric of diversity

The weighted sum method does not guarantee the diversity in solutions over the Pareto front. Even with a proper method for generating weight ratios, the distribution of solutions over P* might vary from one algorithm to another. This can be seen from Fig. 11, where TLBO points (in pink) are all clustered in one area and DE points (in blue) are more evenly distributed across the Pareto front. Another metric can be used to evaluate the diversity of the non-dominated solutions originating from each algorithm on the front in Fig. 10. In other words, this metric estimates how well each algorithm can generate good solutions all along P* with a weighted sum method. The metric of diversity is based on a similar approach of Ref. [66]. The concept of Shannon's entropy (or information theory) is applied to measure how a set of points is spread across a feasible region. A set of solutions with higher entropy means that the solutions have a better coverage of the solution space.

First, the three dimensional feasible domain denoted by F''' is normalized and then subdivided into a grid of $a_1 \times a_2 \times a_3$ cells. The subdivision is determined so that the decision-maker is indifferent to solutions within the same cell. The feasible domain is delimited by the extreme solutions from the approximate Pareto front from Fig. 10. For a given cell, an influence function Ω to each solution is defined. The influence function is a decreasing function of the distance between the specific cell and that solution. In this paper, Ω is defined as a normal distribution by the Gaussian function given as follow:

$$\Omega = \frac{1}{\sigma\sqrt{2\pi}} e^{-0.5\left(r/\sigma\right)^2} \tag{4}$$

where r is the Euclidian distance between the solution and the specified cell and σ is the standard deviation that affect the influence of far solutions on the specific cell. As a general rule, σ is selected subjectively so that solutions at the boundaries from the center of the mesh has no influence (near zero value of Ω).

Then, for each cell, a density function is defined as a collection of the influence functions of all the solution points. The density function D_{ijk} to a specific cell c in the three dimensional solution space is given as:

$$D_{ijk}\left(c\right) = \sum_{s=1}^{N} \Omega_s \left(r_{s\to c}\right) \tag{5}$$

where N is the total number of solutions and $\Omega_s \left(r_{s\to c}\right)$ is the influence function evaluated from the Euclidian distance between the solution s and the cell c.

Afterwards, the density functions for each cells is turned into a normalized density function:

$$\rho_{ijk} = \frac{D_{ijk}}{\sum_{k_1=1}^{a_1}\sum_{k_2=1}^{a_2}\sum_{k_3=1}^{a_3} D_{k_1k_2k_3}}. \tag{6}$$

At last, the entropy H is defined as follow:

$$H = -\sum_{k_1=1}^{a_1}\sum_{k_2=1}^{a_2}\sum_{k_3=1}^{a_3} \rho_{k_1k_2k_3} \ln\left(\rho_{k_1k_2k_3}\right) \tag{7}$$

where $\rho_{k_1k_2k_3} \ln\left(\rho_{k_1k_2k_3}\right)$ is assumed zero when $\rho_{k_1k_2k_3} = 0$. Then, to keep the metric values between 0 and 1, H is normalized by the maximal value obtained from the best algorithm (H_n). One should note that, any solution within a single cell is considered the same. Therefore the division of the feasible area is performed in order to keep the number of solutions within a cell to a minimal while reducing the computational time.

Here, the solution space is subdivided into $20\times20\times20$ cells. A value of σ of 0.01 is selected in this test case.

5.4. Assessment of heuristics based on convergence and diversity metrics

One can see from Fig. 11 that all the algorithms are present on the Pareto front P* in the sense that all algorithms were able to generate non-dominated solutions. However, the number and distribution of the non-dominated points of each algorithm on P* is different. The metrics presented are quantitative tools that provide useful information on the performance of each heuristic algorithm analysed in this paper. For each algorithm, the normalized mean values C_μ calculated from C_n and the normalized entropy H_n are reported in Table 4. From Table 4, the value of the relative mean convergence metric C_μ of each algorithm is roughly similar (values over 0.9), which shows that the front achieved with each heuristic has a similar distance from the allegedly "true" Pareto front. Nonetheless, TLBO stands out from the other algorithms with C_μ of 0.84, which indicates that it is closer to the Pareto front of Fig. 10. However, from Fig. 11, it is visible that the best solutions provided by TLBO are clustered in the lower left corner of the Pareto front, whereas all the other algorithms cover more of the Pareto front. Therefore, from the metric C_μ alone, even if some algorithms like TLBO could appear to be a better choice to reach optimal solutions, the algorithm might not provide the best solutions over the entire solution domain. In contrast, it can be seen from Table 4 that other algorithms, such as DE and ACO_R, have a higher value of C_μ, but are more spread out over the solution domain. [Table 4 near here]

The normalized entropy value H_n also supports this statement. In fact, the normalized metric value H_n of TLBO (0.77) is the lowest while DE and ACO_R have the

highest values (1.00 and 0.98 respectively). In addition, both DE and ACO_R have generated more points on the set P* with 32.7% and 21.3% respectively.

In sum, with the results shown in Table 4, it can be concluded that the five algorithms compared in this work were “functional” and did not yield completely different levels of performance for the test case problem. That being said, DE outperforms the other algorithms in terms of reliability, capacity to reach optimality, and diversity of solutions using the weighted sum method for the multi-objective optimization. However, it has been demonstrated that DE has a slow convergence speed and relatively high setting sensibility on reaching near-optimal solution in comparison to the other algorithms. ACO_R and GA have faster convergence speed, good capacity to reach optimality and good diversity. However, the algorithm parameters of ACO_R are difficult to tune. For easier tuning and adjustments, other algorithm like PSO and TLBO are appealing options. PSO has a fast convergence speed but poor reliability, while TLBO has slow convergence speed but high reliability. However, both algorithms have lower quality and diversity of near-optimal solutions when used with a weighted sum method.

6. Conclusion

The performance of five stochastic global search algorithms with weighted sum approach to solve a multi-objective optimization problem are compared. A nonlinear mixed-variable constrained optimization test case is considered, which consisted in optimizing the design of a thermoelectric device. This comparative analysis is conducted to highlight the forces and weaknesses of each algorithm and help in algorithm selection. The algorithms compared in this paper are: GA, PSO, ACO_R, TLBO and DE.

As a first step, the parameters of each algorithm influencing the intensification and diversification mechanisms have been analyzed to assess their impact on: the capacity to reach optimal solutions, the convergence speed, and on the reliability. TLBO has been removed from this first analysis since the algorithm does not have any specific parameter to fine-tune. From this analysis, it can be concluded that the choice of proper parameters for a specific problem can be influential, although some algorithms are more sensitive to the values of these settings than others. For ACO_R and DE, the solution reached is highly influenced by the choice of parameter values, whereas little effect has been observed on GA and PSO.

As a second step, the weighted sum method is used to solve the multi-objective optimization test case with proper settings. An approximation of the "true" Pareto front is obtained by performing a non-dominated sorting on the set of solutions obtained from every weight ratio combination with all five algorithms. Then, each algorithm is compared using convergence and diversity metrics. In general, all the algorithms are relatively close to the true Pareto front and yield comparable performances. Nonetheless, it has been observed that DE produced more solutions on the Pareto front as well as a highest level of diversity, followed by ACO_R.

The results of this paper provide a useful insight on the selection of heuristic algorithms and their specific settings for similar complex design optimization problems. Future work could investigate other heuristics (e.g., firefly algorithm [67], etc.) and perform this comparison with other types of problems.

Acknowledgements

The authors' work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] L. T. Biegler et I. E. Grossmann, « Retrospective on optimization », *Comput. Chem. Eng.*, vol. 28, nº 8, p. 1169‑1192, juill. 2004.

[2] L. Rios et N. Sahinidis, « Derivative-free optimization: a review of algorithms and comparison of software implementations », *J. Glob. Optim.*, vol. 56, nº 3, p. 1247‑1293, juill. 2013.

[3] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, et A. Mahajan, « Mixed-integer nonlinear optimization*† », *Acta Numer.*, vol. 22, p. 1‑131, mai 2013.

[4] C. Blum et A. Roli, « Metaheuristics in combinatorial optimization: Overview and conceptual comparison », *ACM Comput. Surv. CSUR*, vol. 35, nº 3, p. 268–308, 2003.

[5] C. Oguz et M. F. Ercan, « A Genetic Algorithm for Hybrid Flow-shop Scheduling with Multiprocessor Tasks », *J. Sched.*, vol. 8, nº 4, p. 323‑351, 2005.

[6] A. C. Zecchin, A. R. Simpson, H. R. Maier, et J. B. Nixon, « Parametric study for an ant algorithm applied to water distribution system optimization », *IEEE Trans. Evol. Comput.*, vol. 9, nº 2, p. 175‑191, avr. 2005.

[7] S. Paterlini et T. Krink, « Differential evolution and particle swarm optimisation in partitional clustering », *Comput. Stat. Data Anal.*, vol. 50, nº 5, p. 1220‑1247, mars 2006.

Birattari M (2009) Tuning Metaheuristics: A Machine Learning Perspective, Springer-Verlag, Berlin, Heidelberg

[9] Z. Yuan, M. A. Montes de Oca, M. Birattari, et T. Stützle, « Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms », *Swarm Intell.*, vol. 6, nº 1, p. 49‑75, mars 2012.

[10] R. T. Marler et J. S. Arora, « Survey of multi-objective optimization methods for engineering », *Struct. Multidiscip. Optim.*, vol. 26, nº 6, p. 369‑395, avr. 2004.

[11] I. Das et J. E. Dennis, « A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems », *Struct. Optim.*, vol. 14, nº 1, p. 63‑69, août 1997.

[12] V. Beiranvand, W. Hare, et Y. Lucet, « Best practices for comparing optimization algorithms », *Optim. Eng.*, vol. 18, nº 4, p. 815‑848, déc. 2017.

[14] M. Allyson-Cyr, « Optimisation sous contrainte d'un générateur thermoélectrique pour la récupération de chaleur par différents algorithmes heuristiques », Mémoire, Université Laval, Québec, Canada, 2018.

[15] J. H. Holland, *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*, 1st MIT Press ed. Cambridge, Mass: MIT Press, 1992.

[16] L. Gosselin, M. Tye-Gingras, et F. Mathieu-Potvin, « Review of utilization of genetic algorithms in heat transfer problems », *Int. J. Heat Mass Transf.*, vol. 52, nº 9–10, p. 2169‑2188, avr. 2009.

[17] W. Huang et H. N. Lam, « Using genetic algorithms to optimize controller parameters for HVAC systems », *Energy Build.*, vol. 26, nº 3, p. 277‑282, janv. 1997.

[18] V. K. Mishra, S. C. Mishra, et D. N. Basu, « Simultaneous estimation of parameters in analyzing porous medium combustion—assessment of seven optimization tools », *Numer. Heat Transf. Part Appl.*, vol. 71, nº 6, p. 666‑676, mars 2017.

[19] S. Bélanger et L. Gosselin, « Multi-objective genetic algorithm optimization of thermoelectric heat exchanger for waste heat recovery », *Int. J. Energy Res.*, vol. 36, nº 5, p. 632‑642, avr. 2012.
[20] S. S. Rao et Y. Xiong, « A Hybrid Genetic Algorithm for Mixed-Discrete Design Optimization », *J. Mech. Des.*, vol. 127, nº 6, p. 1100‑1112, oct. 2004.
[21] K. Deep, K. P. Singh, M. L. Kansal, et C. Mohan, « A real coded genetic algorithm for solving integer and mixed integer optimization problems », *Appl. Math. Comput.*, vol. 212, nº 2, p. 505‑518, juin 2009.
[22] K. Deb, A. Pratap, S. Agarwal, et T. Meyarivan, « A fast and elitist multiobjective genetic algorithm: NSGA-II », *IEEE Trans. Evol. Comput.*, vol. 6, nº 2, p. 182‑197, avr. 2002.
[23] J. Kennedy et R. Eberhart, « Particle swarm optimization », in , *IEEE International Conference on Neural Networks, 1995. Proceedings*, 1995, vol. 4, p. 1942‑1948 vol.4.
[24] C. Guo, J. Hu, B. Ye, et Y. Cao, « Swarm intelligence for mixed-variable design optimization », *J. Zhejiang Univ.-Sci. A*, vol. 5, nº 7, p. 851‑860, juill. 2004.
[25] A. P. Silva, M. A. S. S. Ravagnani, E. C. Biscaia, et J. A. Caballero, « Optimal heat exchanger network synthesis using particle swarm optimization », *Optim. Eng.*, vol. 11, nº 3, p. 459‑470, sept. 2010.
Yousefi M, Yousefi M, Martins Ferreira RP, Darus AN (2017) A swarm intelligent approach for multi-objective optimization of compact heat exchangers, Proc Inst Mech Eng Part E J Process Mech Eng 231:164–171. doi: 10.1177/0954408915581995
[27] A. Ibrahim, S. Rahnamayan, M. Vargas Martin, et B. Yilbas, « Multi-objective thermal analysis of a thermoelectric device: Influence of geometric features on device characteristics », *Energy*, vol. 77, p. 305‑317, déc. 2014.
[28] D. R. O. G, L. A. L. de Almeida, et O. A. C. Vilcanqui, « Parameter identification of thermoeletric modules using particle swarm optimization », in *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, 2015, p. 812‑817.
[29] Udayraj, K. Mulani, P. Talukdar, A. Das, et R. Alagirusamy, « Performance analysis and feasibility study of ant colony optimization, particle swarm optimization and cuckoo search algorithms for inverse heat transfer problems », *Int. J. Heat Mass Transf.*, vol. 89, nº Supplement C, p. 359‑378, oct. 2015.
[30] A. Bangian-Tabrizi et Y. Jaluria, « A study of transient wall plume and its application in the solution of inverse problems », *Numer. Heat Transf. Part Appl.*, vol. 75, nº 3, p. 149‑166, févr. 2019.
[31] S. He, E. Prempain, et Q. H. Wu, « An improved particle swarm optimizer for mechanical design optimization problems », *Eng. Optim.*, vol. 36, nº 5, p. 585‑605, oct. 2004.
[32] J. Clarke, L. McLay, et J. T. McLeskey, « Comparison of genetic algorithm to particle swarm for constrained simulation-based optimization of a geothermal power plant », *Adv. Eng. Inform.*, vol. 28, nº 1, p. 81‑90, janv. 2014.
[33] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. C. Coello, F. Luna, et E. Alba, « SMPSO: A new PSO-based metaheuristic for multi-objective optimization », in *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making(MCDM)*, 2009, p. 66‑73.

[34] M. Clerc et J. Kennedy, « The particle swarm - explosion, stability, and convergence in a multidimensional complex space », *IEEE Trans. Evol. Comput.*, vol. 6, nº 1, p. 58‑73, févr. 2002.
[35] Z. H. Zhan, J. Zhang, Y. Li, et H. S. H. Chung, « Adaptive Particle Swarm Optimization », *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 39, nº 6, p. 1362‑1381, déc. 2009.
[36] R. Eberhart et J. Kennedy, « A new optimizer using particle swarm theory », in , *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS '95*, 1995, p. 39‑43.
[37] M. DORIGO, « Optimization, Learning and Natural Algorithms », *PhD Thesis Politec. Milano Italy*, 1992.
[38] M. Dorigo, G. Di Caro, et L. M. Gambardella, « Ant Algorithms for Discrete Optimization », *Artif. Life*, vol. 5, nº 2, p. 137‑172, Spring 1999.
[39] E. Hetmaniok, D. Słota, et A. Zielonka, « Determination of the Heat Transfer Coefficient by Using the Ant Colony Optimization Algorithm », in *Parallel Processing and Applied Mathematics*, 2011, p. 470‑479.
[40] B. Zhang, H. Qi, Y.-T. Ren, S.-C. Sun, et L.-M. Ruan, « Application of homogenous continuous Ant Colony Optimization algorithm to inverse problem of one-dimensional coupled radiation and conduction heat transfer », *Int. J. Heat Mass Transf.*, vol. 66, nº Supplement C, p. 507‑516, nov. 2013.
[41] I. C. Silva, F. R. do Nascimento, E. J. de Oliveira, A. L. M. Marcato, L. W. de Oliveira, et J. A. Passos Filho, « Programming of thermoelectric generation systems based on a heuristic composition of ant colonies », *Int. J. Electr. Power Energy Syst.*, vol. 44, nº 1, p. 134‑145, janv. 2013.
[42] B. Zhang, H. Qi, S.-C. Sun, L.-M. Ruan, et H.-P. Tan, « A novel hybrid ant colony optimization and particle swarm optimization algorithm for inverse problems of coupled radiative and conductive heat transfer », *Therm. Sci.*, vol. 20, nº 2, p. 461‑472, 2016.
[43] H. Qi, B. Zhang, S. Gong, et L.-M. Ruan, « Simultaneous retrieval of multiparameters in a frequency domain radiative transfer problem using an improved pdf-based aco algorithm », *Numer. Heat Transf. Part Appl.*, vol. 69, nº 7, p. 727‑747, avr. 2016.
[44] K. Socha et M. Dorigo, « Ant colony optimization for continuous domains », *Eur. J. Oper. Res.*, vol. 185, nº 3, p. 1155‑1173, mars 2008.
[45] T. Liao, K. Socha, M. A. M. de Oca, T. Stützle, et M. Dorigo, « Ant Colony Optimization for Mixed-Variable Optimization Problems », *IEEE Trans. Evol. Comput.*, vol. 18, nº 4, p. 503‑518, août 2014.
[46] A. E. L. Rivas et L. A. G. Pareja, « Coordination of directional overcurrent relays that uses an ant colony optimization algorithm for mixed-variable optimization problems », 2017, p. 1‑6.
[47] R. V. Rao, V. J. Savsani, et D. P. Vakharia, « Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems », *Comput.-Aided Des.*, vol. 43, nº 3, p. 303‑315, mars 2011.
[48] R. V. Rao, V. J. Savsani, et D. P. Vakharia, « Teaching–Learning-Based Optimization: An optimization method for continuous non-linear large scale problems », *Inf. Sci.*, vol. 183, nº 1, p. 1‑15, janv. 2012.

[49] R. V. Rao, V. J. Savsani, et J. Balic, « Teaching–learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems », *Eng. Optim.*, vol. 44, nº 12, p. 1447‑1462, déc. 2012.
[50] R. V. Rao et V. Patel, « Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm », *Appl. Math. Model.*, vol. 37, nº 3, p. 1147‑1162, févr. 2013.
[51] R. Venkata Rao et V. Patel, « Multi-objective optimization of two stage thermoelectric cooler using a modified teaching–learning-based optimization algorithm », *Eng. Appl. Artif. Intell.*, vol. 26, nº 1, p. 430‑445, janv. 2013.
[52] R. V. Rao et K. C. More, « Optimal design of the heat pipe using TLBO (teaching–learning-based optimization) algorithm », *Energy*, vol. 80, p. 535‑544, févr. 2015.
[53] F. Zou, L. Wang, X. Hei, D. Chen, et B. Wang, « Multi-objective optimization using teaching-learning-based optimization algorithm », *Eng. Appl. Artif. Intell.*, vol. 26, nº 4, p. 1291‑1300, avr. 2013.
[54] R. Storn et K. Price, « Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces », *J. Glob. Optim.*, vol. 11, nº 4, p. 341‑359, déc. 1997.
[55] B. V. Babu et S. A. Munawar, « Differential evolution strategies for optimal design of shell-and-tube heat exchangers », *Chem. Eng. Sci.*, vol. 62, nº 14, p. 3720‑3739, juill. 2007.
[56] J. Chen, G. Cui, et H. Duan, « Multipopulation differential evolution algorithm based on the opposition-based learning for heat exchanger network synthesis », *Numer. Heat Transf. Part Appl.*, vol. 72, nº 2, p. 126‑140, juill. 2017.
[57] F. Neri et V. Tirronen, « Recent advances in differential evolution: a survey and experimental analysis », *Artif. Intell. Rev.*, vol. 33, nº 1/2, p. 61‑106, févr. 2010.
[58] S. Qian, Y. Ye, Y. Liu, et G. Xu, « An improved binary differential evolution algorithm for optimizing PWM control laws of power inverters », *Optim. Eng.*, vol. 19, nº 2, p. 271‑296, juin 2018.
[59] S. Kukkonen et J. Lampinen, « GDE3: the third evolution step of generalized differential evolution », in *2005 IEEE Congress on Evolutionary Computation*, 2005, vol. 1, p. 443-450 Vol.1.
[60] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, et T. Stützle, « The irace package: Iterated racing for automatic algorithm configuration », *Oper. Res. Perspect.*, vol. 3, p. 43‑58, 2016.
[61] F. Hutter, H. H. Hoos, et K. Leyton-Brown, « Sequential Model-Based Optimization for General Algorithm Configuration », in *Learning and Intelligent Optimization*, vol. 6683, C. A. C. Coello, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, p. 507‑523.
[62] C. Preechakul et S. Kheawhom, « Modified genetic algorithm with sampling techniques for chemical engineering optimization », *J. Ind. Eng. Chem.*, vol. 15, nº 1, p. 110‑118, janv. 2009.
[63] R. K. Ursem et P. Vadstrup, « Parameter identification of induction motors using stochastic optimization algorithms », *Appl. Soft Comput.*, vol. 4, nº 1, p. 49‑64, févr. 2004.
[64] J. Vesterstrom et R. Thomsen, « A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems », in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, 2004, vol. 2, p. 1980-1987 Vol.2.

[65] K. Deb et S. Jain, « Running performance Metrics for evolutionary multi-objective optimization », 2002.
[66] A. Farhang-Mehr et S. Azarm, « Diversity assessment of Pareto optimal solution sets: an entropy approach », in *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC '02*, 2002, vol. 1, p. 723‑728.
[67] L. F. F. Miguel, L. F. Fadel Miguel, et R. H. Lopez, « A firefly algorithm for the design of force and placement of friction dampers for control of man-induced vibrations in footbridges », *Optim. Eng.*, vol. 16, nº 3, p. 633‑661, sept. 2015.

Table 1: Discrete and continuous decision variables of the TEG model (with 12 × 12 grid), based on Ref. [14]

Variables	Number of variables	Bound
Thermoelectric material selection	144	{1…3}
Thermoelectric electric connection grid layout	1	{1…24}
Heat collector flow grid layout	1	{1…24}
Heat collector duct geometry	1	{1,2}
Number of fins (for each duct)	12	{0…100}
Total electric current [A]	1	[0.1…100]
Surface-to-length ratio of modules [m]	3	[0.0001…0.01]
Global heat transfer coefficient U [W/m^2K]	1	[1…100]
Capacity rate of the collector C [W/K]	1	[100…25,000]

Table 2: List and range of algorithm parameters

Algorithm	Parameters	Combination tested
GA	CR	[0 : 0.05 : 1]
	Crossover fct.	Scattered - single point - double points
	Selection fct.	uniform - roulette wheel - Tournament
PSO	$w_{inertia}$	1
	w_{damp}	[0.7 ; 0.99 ; 1]
	c_1	[0 : 0.2 : 2]
	c_2	[0 : 0.2 : 2]
ACO	q	[0.001 : 0.001 : 0.02]
	ζ	[0.1 : 0.1 : 1]
DE	β	[0.01 : 0.01 : 0.1]
	pCR	[0.1 : 0.1 : 1]
TLBO	-	-

Table 3: Best parameters for nearly optimal net power output

Alg.	Best parameters		$\max(P_{net})$ [W]	*RSD*
GA	% Crossover	0.70	11,936	0.0173
	Crossover fct.	Tournament		
	Selection fct.	Scattered		
PSO	$w_{inertia}$	1	11,928	0.0217
	w_{damp}	0.99		
	c_1	0.6		
	c_2	1		
ACO_R	q	0.006	11,977	0.0109
	ζ	0.4		
DE	β	0.02	12,004	0.0091
	pCR	0.8		
TLBO	-	-	11,817	0.0083

Table 4: Normalized mean metric value C_μ, normalized entropy H_n and percentage of points that belongs to the approximate Pareto front P^*

Algorithm	C_μ	H_n	$\%\,pts$
GA	0.97	0.96	17.9
PSO	0.93	0.84	10.4
ACO_R	0.97	0.98	21.3
DE	0.99	1.00	32.7
TLBO	0.85	0.77	17.7

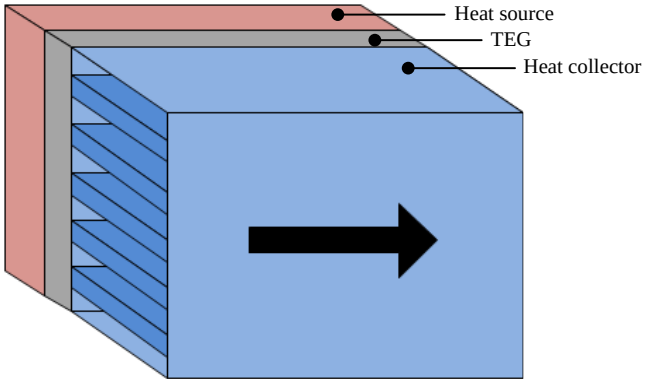

Figure 1

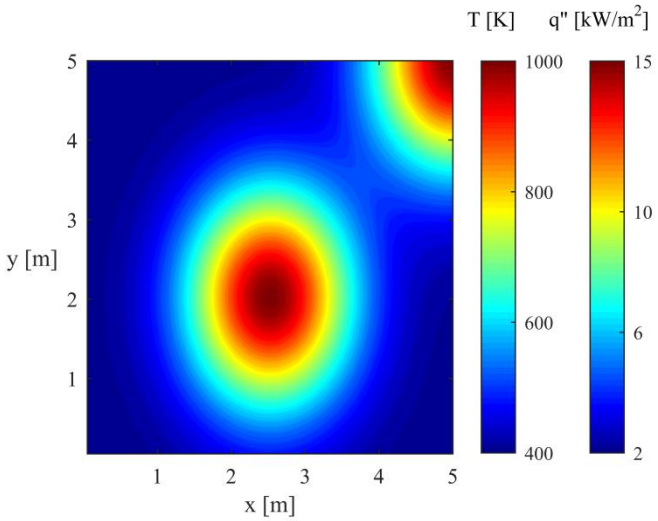

Figure 2

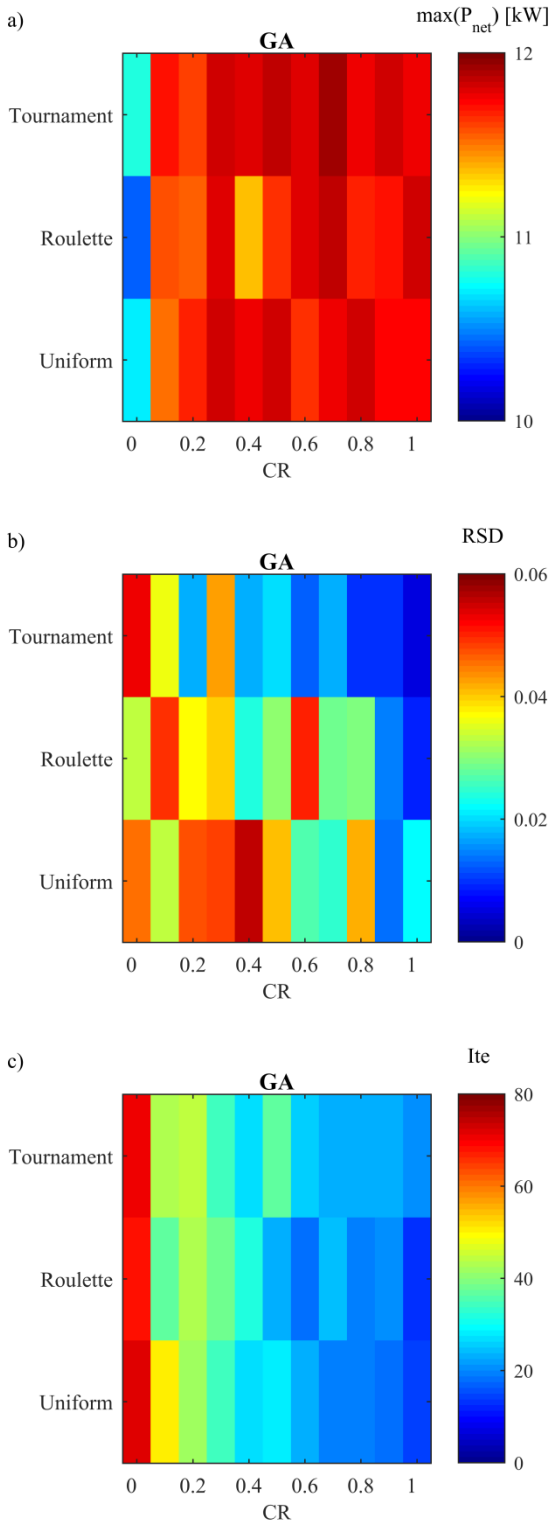
a)
max(P_net) [kW]
GA
Tournament
Roulette
Uniform
0
0.2
0.4
0.6
0.8
1
CR
12
11
10
b)
RSD
GA
Tournament
Roulette
Uniform
0
0.2
0.4
0.6
0.8
1
CR
0.06
0.04
0.02
0
c)
Ite
GA
Tournament
Roulette
Uniform
0
0.2
0.4
0.6
0.8
1
CR
80
60
40
20
0

Figure 3

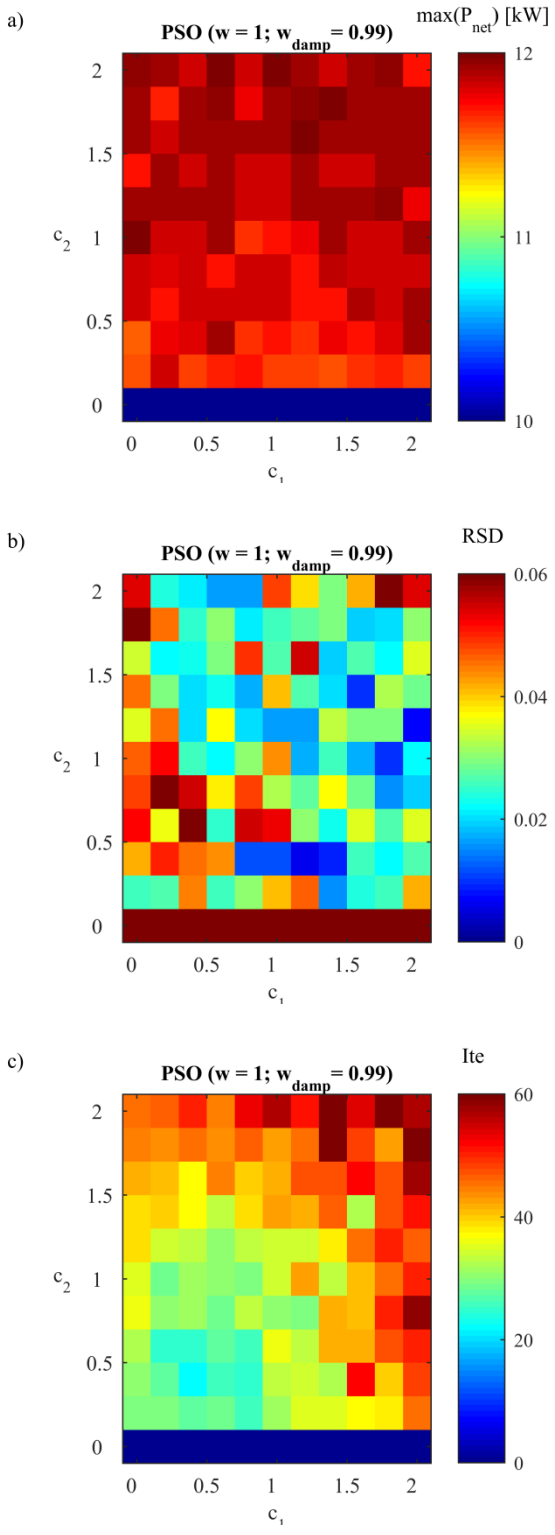
a)
PSO (w = 1; w_damp = 0.99)
max(P_net) [kW]
12
11
10
b)
PSO (w = 1; w_damp = 0.99)
RSD
0.06
0.04
0.02
0
c)
PSO (w = 1; w_damp = 0.99)
Ite
60
40
20
0
c2
c1
0
0.5
1
1.5
2

Figure 4

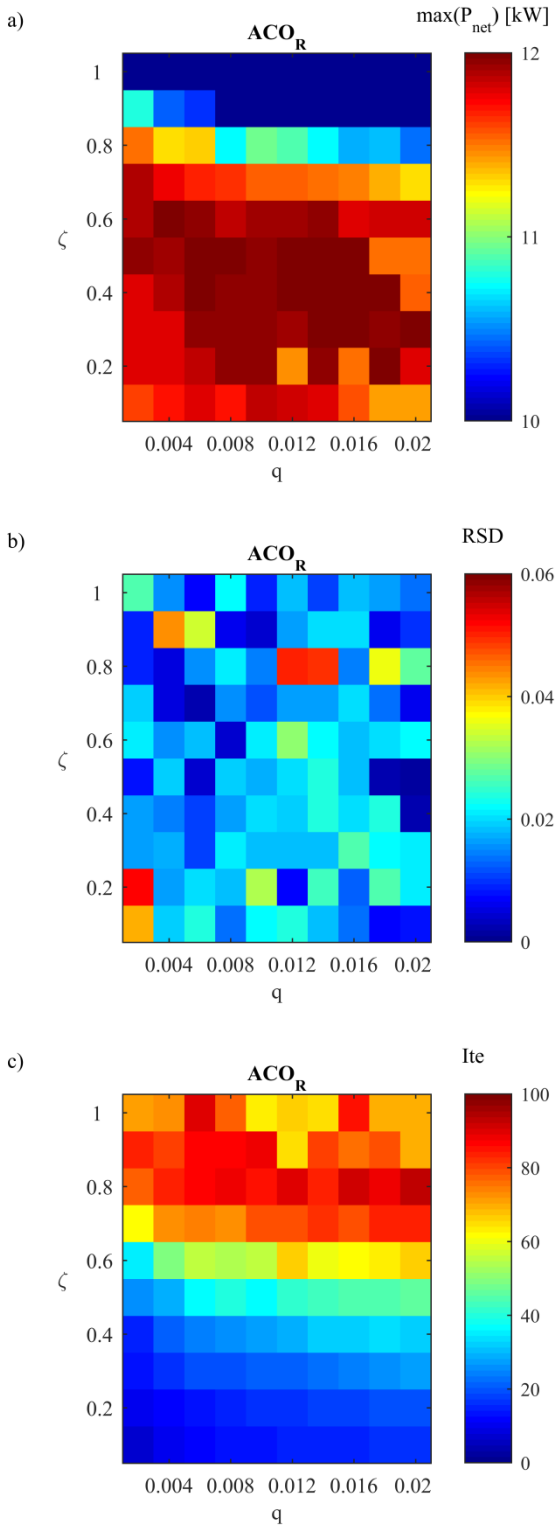
a)
ACO_R
max(P_net) [kW]
1
0.8
0.6
ζ
0.4
0.2
0.004 0.008 0.012 0.016 0.02
q
12
11
10
b)
ACO_R
RSD
0.06
0.04
0.02
0
c)
ACO_R
Ite
100
80
60
40
20
0

Figure 5

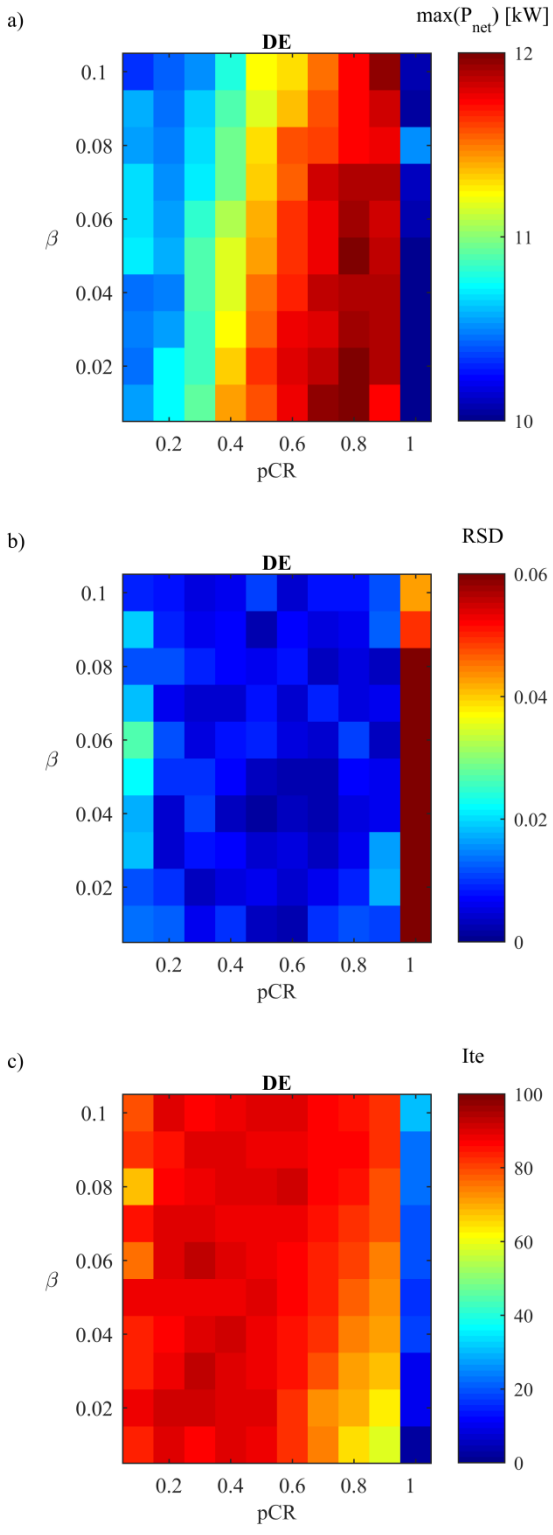
a)
max(P_net) [kW]
DE
0.1
0.08
0.06
0.04
0.02
β
0.2
0.4
0.6
0.8
1
pCR
12
11
10
b)
RSD
DE
0.06
0.04
0.02
0
c)
Ite
DE
100
80
60
40
20
0

Figure 6

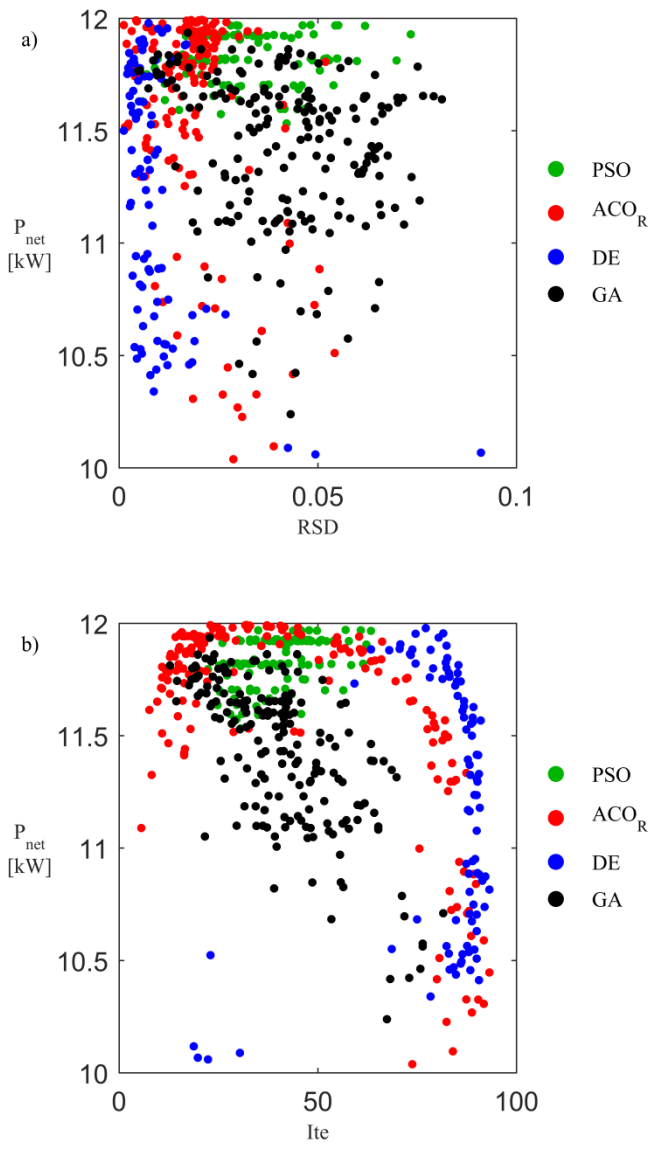
a)
12
11.5
11
10.5
10
P_{net} [kW]
0
0.05
0.1
RSD
PSO
ACO_R
DE
GA
b)
12
11.5
11
10.5
10
P_{net} [kW]
0
50
100
Ite
PSO
ACO_R
DE
GA

Figure 7

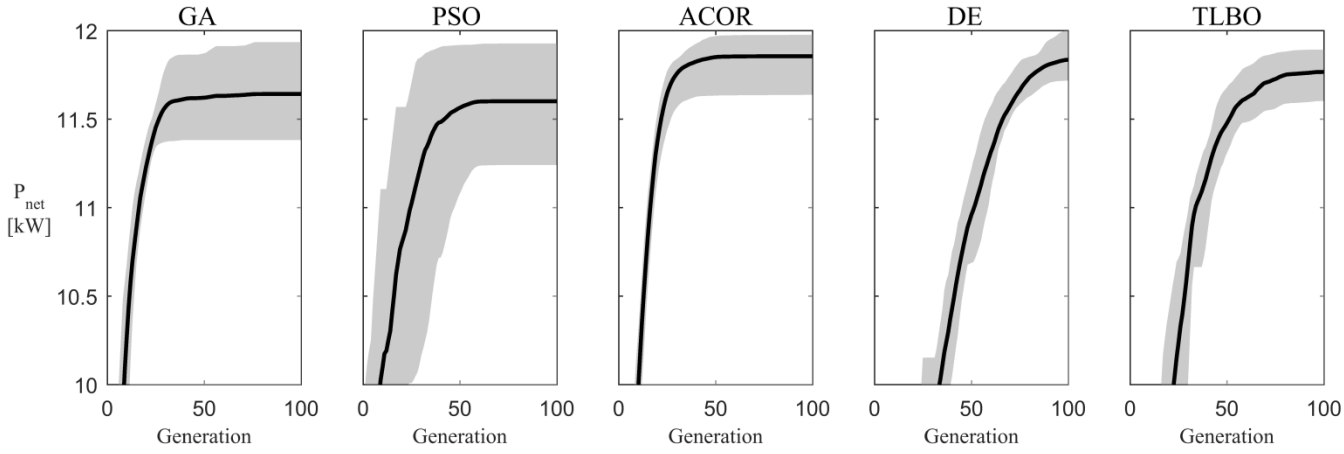

Figure 8

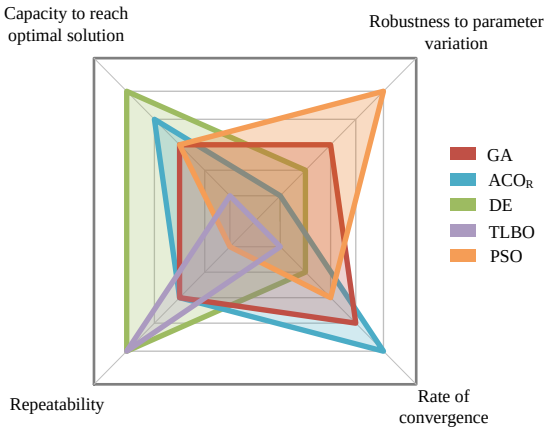
Capacity to reach optimal solution
Robustness to parameter variation
GA
ACOR
DE
TLBO
PSO
Repeatability
Rate of convergence

Figure 9

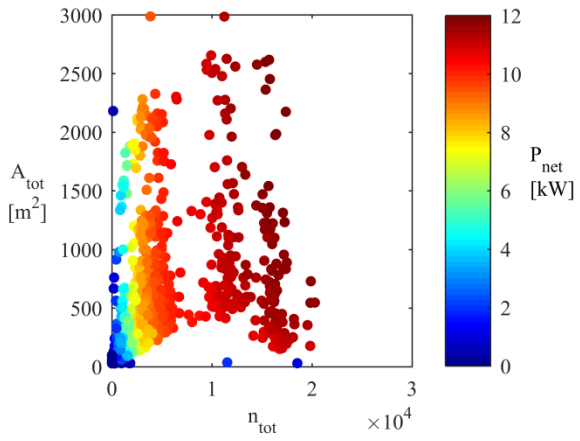
3000
2500
2000
1500
1000
500
0
A tot
[m2]
0
1
2
3
n tot
×10^4
12
10
8
6
4
2
0
P net
[kW]

Figure 10

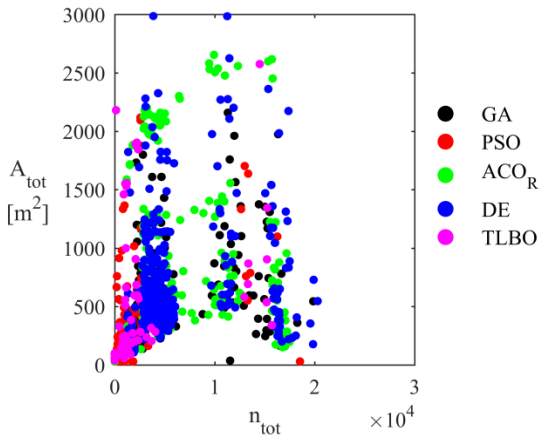
3000
2500
2000
1500
1000
500
0
A tot
[m2]
0
1
2
3
n tot
×10^4
GA
PSO
ACO R
DE
TLBO

Figure 11

Figure Captions

Figure 1: Schematic view of the TEG system sandwiched between the heat source and a heat collector.

Figure 2: Constrained temperature and heat flux distributions over the heat source surface.

Figure 3: Impact of GA parameters on a) maximal net power, c) RSD and c) Ite.

Figure 4: Impact of PSO parameters on a) maximal net power, c) RSD and c) Ite.

Figure 5: Impact of ACOR parameters on a) maximal net power, c) RSD and c) Ite.

Figure 6: Impact of DE parameters on a) maximal net power, c) RSD and c) Ite.

Figure 7: Effect of the algorithm parameters on a) RSD and b) the required number of iterations using GA, PSO, ACOR, and DE for all the settings tested.

Figure 8: Convergence comparison of the net power optimization with algorithm parameters of Table 3. The lines are the mean value and the gray areas are the min-max range.

Figure 9: Qualitative summary of the heuristic algorithm performance for the problem tested.

Figure 10: Approximate Pareto front P* after the non-dominated sorting obtained with the sum-weight method with all solutions from GA, PSO, ACOR, TLBO and DE.

Figure 11: Origin of the non-dominated solutions achieved by all algorithms.