Cooperation-based search of global optima

Damien Vergnet[0000−0003−2948−8807], Elsy Kaddoum, Nicolas Verstaevel, Frédéric Amblard

IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, **UT1**, **UT2**, Toulouse, France <firstname>.<lastname>@irit.fr

Abstract. A new cooperation-based metaheuristic is proposed for searching gobal optima of functions. It is based on the assumption that the dynamics of the objective function does not change significantly between iterations. It relies on a local search process coupled with a cooperative semi-local search process. Its performances are compared against four other metaheuristics on unconstrained mono-objective optimization problems. Results show that the proposed metaheuristic is able to find the global minimum of the tested functions faster than the compared methods while reducing the number of iterations and the number of calls of the objective function.

Keywords: local cooperation · collective decision · metaheuristic optimization · local search

1 Introduction

The simulation of systems is a powerful tool to understand their behaviors and underline their advantages and limits. Several studies aim at reconstructing virtual systems called digital twins to simulate and verify the behavior of specific systems. Such systems can be used in mobility or natural disaster studies to reproduce specific simulation conditions and understand the reasons of such phenomena [4]. Building a digital twin that reproduces the exact behavior of a real system is not an easy task. As real systems are generaly complex systems with non-linear interdependencies among their parameters, finding the best modeling functions and adapting in real-time their parameters to keep a simulation close to the real behavior of the system is not trivial. Many studies have formalised the calibration problem as an optimization problem where the parameters of the modeling functions are tuned by optimizing an objective function: simulation parameters become decision variables and relevent model outputs are integrated into objective functions [2,8]. This implies the need for a fast optimization system that is able to rapidly adapt to changes that may occur in the real system.

Multiple optimization methods exist that could be used to solve this problem but they present important drawbacks such as a tendency to converge towards local optima or are too slow [6,11,14].

In this paper we propose a new metaheuristic local optimization method named **CoBOpti**, which stands for **Co**operation-**B**ased **Opti**mization. It is

based on an hypothesis of local continuity of the objective function, i.e. the value of the objective function does not vary dramatically when the value of decisions variables varies little. Compared to standard state of the art methods, CoBOpti reaches optimal solutions while reducing the number of iterations and objective function evaluation.

The main contributions of this paper are as follows:

- We introduce a **new local optimization metaheuristic based on an hypothesis of local continuity and cooperation**. This hypothesis allows to model the problem of searching for a global optimum as **a cooperation problem** where a point determines the next point to explore by exploiting the information of its neighbours.
- We experiment and compare our approach on unconstrained mono-objective optimization problems with a single decision variable to demonstrate that **the proposed approach allows to reach a global optimum while minimizing the number of evaluation of the objective function**.

The paper is organised as follow : section 2 discusses the limitations of existing metaheuristics. Section 3 presents our approach and how it gives an answer to these limitations. In section 4, we introduce the results of our experimentation, which is then discussed in section 5 before concluding with limitations and suggest further research.

2 Literature Review

Optimization problems are defined by [3] as finding a vector $\bar{x}_n^* = (x_1^*, ..., x_n^*)$ that optimizes an objective function

$$\bar{f}_k(\bar{x}_n) = (o_1(\bar{x}_n), ..., o_k(\bar{x}_n)) \tag{1}$$

where $\bar{x}_n = (x_1, ..., x_n)$ is a vector of n decision variables.

Many methods exist to solve optimization problems, each making some assumptions on the nature of the problem. One category of such optimization methods is called metaheuristics. [6] defines metaheuristics as methods that perform local and higher level search procedures that are capable of escaping local optima. This definition notably includes methods that employ the notion of neighborhood. The neighborhood of a solution s is the set of all solutions that can be reached from s.

Metaheuristics are interesting for solving optimization problems as they are designed to efficiently explore complex search spaces [6]. Sörensen *et al.* [12] further state that the large majority of real-life optimization problems are more easily solved by metaheuristics, hence our focus on these methods in this paper.

Metaheuristics rely on two important notions: **intensification** and **diversification**. Intensification is a process through which portions of the search space that seem "promising" are explored more thoroughly, i.e. in the neighborhood of the best solutions found yet. Diversification, on the other hand, is a process

aimed at exploring unexplored parts of the search space in hopes to find better solutions. It usually relies on a some form of memory of visited solutions [5].

There are numerous metaheuristics, each with their own hypotheses. As the goal of our proposition is to be used to perform on-line calibration, it needs to rely on fast algorithms and and to be able to handle the set of visited solutions. The presented methods are thus focused around local search and population-based meta-heuristics.

Local search algorithms explore the search space by exploring the immediate neighborhood of the current solution s and selecting the neighbor solution that has a lower objective value than s. In order to escape from local optima, they feature some sort of hill-climbing process that allows degrading the objective value. Such methods include Simulated Annealing (SA), Generalized Simulated Annealing (GSA), Iterated Local Search, Guided Local Search, etc. [6]. The main advantage of these methods is their rapidity, but an important limitation is their tendency to get stuck in local optima [11]. Somes types of local search metaheuristics rely on some kind of memory of visited solutions to try circumvent this limitation such as Tabu Search [6].

Another category of metaheuristics is the **population-based algorithms**. These methods rely on a set of solutions, called the population. The search space is explored by evaluating each solution and modifying them using a set of simple rules. There are two sub-groups in this category: evolutionary and other nature-inspired methods.

Evolutionary algorithms (EA) are iterative methods centered around the notion of *fitness*. The fitness of a solution represents the quality of this solution based on the objective function. During each iteration, called a *generation*, the fitness of each solution is evaluated. Solutions that feature a high enough fitness value are kept for the next generation, all other are discarded. New solutions are generated by stochasticaly crossing over and modifying (mutating) the solutions that were kept after the selection process. This category includes methods such as Genetic Algorithms, Differential Evolution (DE) and Genetic Programming [6,9]. Contrary to local search methods, EAs explore the search space more thoroughly with bigger population sizes and thus are a lot less susceptible to get stuck in local optima. However, they require more computing power and show slower resolution times.

Other population-based methods behave differently from EAs. They still rely on a set of solutions but draw inspiration from complex biological systems such as bird flocking or ant colonies. They feature the same advantage as EAs, i.e. a more thorough exploration of the search space than local search, but still suffer from the same drawbacks of longer computation times and high computing power requirements [6]. Some methods such as Particle Swarm Optimization (PSO) also suffer from a tendency to converge towards local optima because of a poor distribution of information in the population [14].

In our method we propose to combine the speed of local search approaches and the distribution of information of population-based methods. To achieve this goal we borrow the notions of neighborhood and collective reasoning from

these methods. Based on the assumption that **the dynamics of the objective function do not change significantly between two very close points**, we propose a system that **searches for a global optimum through the collective reasoning of already visited solutions**.

Local search and population-based metaheuristics were presented with some of their limitations in the context of optimization for on-line calibration. The next section describes our method, CoBOpti, which is evaluated in section 4.

3 CoBOpti: Cooperation-Based Optimization

In this section, we introduce CoBOpti, a Cooperation-Based Optimization metaheurtistic. The method we propose combines the advantages of both local search and population-based algorithms: the speed of the former and the information distribution of the latter.

Section 3.1 describes the general principle of the approach by giving an overview of the different search phases; section 3.2 details the local search process; section 3.3 details the semi-local search process and how it enables getting out of local minima; finally, section 3.4 describes how points cooperate to solve specific situations.

3.1 General Principle

The goal of CoBOpti is to iteratively explore the surface of an objective function in order to reach a global optimum. During each iteration, the system has to determine the next point to explore. A **point** p_i is defined as a pair $p_i = (x_i, o_i)$ where x_i is the value of the single decision variable and o_i is the value of the objective function at x_i. The succession of visited points is called a **chain**. The algorithm is composed of 4 phases (Figure 1).

The algorithm combines two different heursitics: a **local** one (**phases 1, 2 and 3**), which objective is to discover a local minimum, and **semi-local** one (**phase 4**), which uses the set of local minimum already discovered to look for a global minimum.

The goal of **local search** (**phase 1**) is to find a local minimum. Each iteration t starts with a chain containing some already visited points $p(t)$, $p(t-1)$, etc. Among all the points in the chain, the system choose two points to determine in which direction it needs to go (**phases 2 and 3**). This process continues until a local minimum has been found, i.e. the distance along the x axis between the two points with the lowest objective value of the chain is less than ε_{dist}.

The objective of **semi-local search** is to explore the function towards a global minimum. This process has to decide which point $p(t+1)$ to explore based on already visited local minima (**phase 4**). Every time the semi-local search has decided on which point to explore next, a new chain is created and the local search continues from this new point.

The search stops when a visited local minimum has an objective value less than a predefined threshold ε_{obj}.

The notion of chains is important as it isolates clusters of points (black and red dots in figure 1). It is not desirable that distant points interact during the local search process because of potential higher discrepencies between the actual function value and its estimation. Using chains implies that distant points cannot be used together to compute linear approximations during local search and thus mitigates potential errors. Several chains are created during the optimization process.

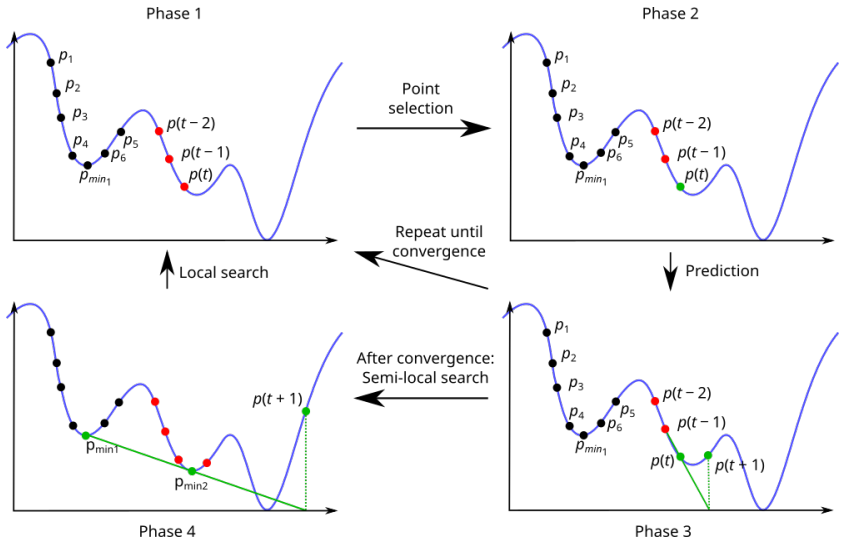

Fig. 1. The search phases of CoBOpti: point selection, local search, higher level search

The following sections detail how points are selected and how $p(t+1)$ is computed. Section 3.2 describes how the local search process selects points to reach a local minima; section 3.3 describes how the system gets out of local minima and searches for a global optimum; finally, section 3.4 describes how points cooperate to solve some difficult situations.

3.2 Local Search

The objective of local search is to follow the curve of the objective function to find a local minimum. At each iteration t, the next point $p(t+1)$ to explore is determined by computing linear approximations of the objective function using two points of the current chain.

Therefore, at each iteration t, two points need to be selected among those in the current chain. The first selected point is the one with the lowest objective value of the chain at time t, noted p_{min}. The second selected point is one of the neighbors of p_{min}. Two points p_1 and p_2 of a chain are said to be **neighbors** if

they are immediately next to each other, i.e. there is no third point p_3 between them along the x axis. A point can have a maximum of two neighbors. For example, in figure 1, points p_1 and p_2 are neighbors but points p_2 and p_4 are not.

As p_{min} is the point with the lowest objective value of its chain, it has either one or two neighbors at any given time.

Phases 2 and 3 of figure 1 illustrate the first situation, where $p(t) = p_{min}$ (green point) has a single neighbor $p(t-1)$. The x component of the next point $p(t+1)$ is computed by a linear approximation of the objective function between $p_{min} = (x_{min}, o_{min})$ and its only neighbor $p(t-1) = p_n = (x_n, o_n)$:

$$x(t+1) = x_n + \frac{-o_n(x_{min} - x_n)}{o_{min} - o_n} \tag{2}$$

This equation returns the x component of the point that would have an objective value of 0 according to the linear approximation of the objective function.

To ensure that the initial assumption on the function's dynamics stays true, the next point cannot be farther than k_{dist} times the distance between p_{min} and p_n. If it is the case, $x(t+1)$ is set to $x_{min} + k_{dist}(x_{min} - x_n)$. In our experiments, $k_{dist} = 5$ was used.

In the second situation, where p_{min} has two neighbors p_l and p_h, as p_{min} is the point with the lowest known objective value, both neighbors have a higher objective value. This implies that a local minimum is somewhere between p_l and p_h. $x(t+1)$ is thus determined by:

$$x(t+1) = \frac{x_{min} + x_n}{2} \tag{3}$$

where x_n is the x component of either p_l or p_h alternatively. Figure 1 shows an example of this situation (black points). The point p_6 was computed this way, using points p_4 as p_{min} and p_5 as its lowest neighbor.

It should be noted that the objective function value does not need to be re-evaluated at the location of the selected neighbor as it is assumed that it has not changed since it was first evaluated.

This whole process repeats until a local minimum is found. The point p_{min} is considered to be a local minimum when the distance to one of its neighbors is less than ε_{dist}.

3.3 Semi-Local Search

The goal of the semi-local search is to find a global minimum. The way points are selected is similar to what was described in the local search process but differs in some key aspects.

In order to compute x component of the next point $p(t+1)$ using linear approximations of the objective function, two points are selected: the latest local minimum $p_{min1} = (x_{min1}, o_{min1})$ found by the local search process and one of its neighbors. The **neighbors** of a local minimum are the other adjacent local

minima. As with regular points described in section 3.2, local minima have a maximum of two neighbors.

The selected local minimum can have one or two neighbors. Table 1 describes which neighbor is selected depending on the precise situation, where $p_l = (x_l, o_l)$ (resp. $p_h = (x_h, o_h)$) are neighbors of p_{min1} with a lower (resp. higher) x value.

Table 1. Selected neighbor of p_{min1} depending on the situation

	Situation	Selected neighbor
1	One neighbor p_n	p_n
2	Two neighbors, $o_l < o_{min1} < o_h$	p_l
3	Two neighbors, $o_l > o_{min1} > o_h$	p_h
4	Two neighbors, $o_l < o_{min1}$ and $o_{min1} > o_h$	p_l if $o_l < o_h$, otherwise p_h
5	Two neighbors, $o_l > o_{min1}$ and $o_{min1} < o_h$	p_l if $o_l < o_h$, otherwise p_h

For situations 1, 2, 3 and 4, the next point $x(t+1)$ is computed using equation 2, swapping p_{min} for p_{min1} and $p(t-1)$ for the selected neighbor. Phase 4 of figure 1 illustrates this process for situation 1. In this diagram, there are two known local minima, p_{min1} and p_{min2}, the latter being the newly found one. The next point $p(t+1)$ is estimated using a linear approximation between both local minima. As with the local search, $p(t+1)$ cannot be farther than $k|x_{min1} - x_n|$, if it is the case, the same operations are applied as described in section 3.2.

In situation 5, as both neighbors p_l and p_h of p_{min1} have a higher objective value, a global minimum is probably between p_l and p_h. Equation 3 is used again to determine the next point.

Once $x(t+1)$ has been computed, the local search process resumes from this new point with a new chain.

3.4 Cooperation Mechanisms

Sections 3.2 and 3.3 described the nominal behavior of CoBOpti. The system may encounter a number of special situations during both local and semi-local searches. This section presents cooperation rules to detect and solve them.

Case 1. During local search, when a new chain is created, either because it is the first iteration or the semi-local search created a new one, there is a single point inside the chain. This point thus has no neighbors to compute the next point with. Hence, no linear approximation can be estimated and $x(t+1)$ is directly chosen randomly among $\{x_{min}-\delta, x_{min}+\delta\}$ where $\delta = \frac{1}{k_{prop}}|x_{low}-x_{high}|$ and x_{low} (resp. x_{high}) the lower (resp. higher) bounds of the definition domain of x. In our experiment, $k_{prop} = 100$ was used.

Case 2. During semi-local search, a similar situation may occur where there is only one known local minimum. As there are no neighbors to make linear approximations with, a **hill-climbing** process is initiated to escape the local minimum. This process relies on the two points of the latest chain that have the

lowest and highest x value, called extrema. The goal is to climb up the slopes around the local minimum to find another slope of opposite direction.

The search focuses on the slope where the extremum with the lowest objective value is. The next point is computed using equation 4 where $p_e^l = (x_e^l, o_e^l)$ is the extremum with the lowest objective value and $p_e^h = (x_e^h, o_e^h)$ is the other. $p_n = (x_n, o_n)$ is the neighbor of p_e^l. This equation computes the x value of the next point which would have an objective value equal to that of the highest extremum, according to the linear approximation of the objective function between the lowest extremum and its neighbor.

$$x(t+1) = x_e^l + \frac{(o_e^h - o_e^l)(x_n - x_e^l)}{o_n - o_e^l} \tag{4}$$

At the next iteration, if the actual objective value is higher than o_e^h, the process switches sides; if this is not the case, it continues as is. This process is repeated until the actual objective value is lower than o_e^l. The local search process then resumes with a new chain.

During this hill-climbing phase, the distance $|x(t+1) - x_e^l|$ cannot be smaller than a threshold δ_{min} in order to prevent the process from slowing down too much.

Case 3. It may happen that the local search process finds a local minimum that was already discovered in previous iterations. In order to escape a potential search loop, two decisions may occur. If a hill-climbing phase was previously initiated at this local minimum, the next point $x(t+1)$ is computed again and multiplied by a factor of 2, to explore twice as far and explore a new area. On the contrary, if no hill-climbing phase was ever initiated at this local minimum, one is started, in hopes to find a new adjacent valley.

Two local minima are considered to be identical if their distance along the x axis is less than a threshold ε_{same}.

In this section we presented our approach. It relies on the notion of chains of points. We first presented a local search process on a chain that allows finding local optima. When a local optimum is found, a semi-local search process allows finding new regions of the search-space to explore. Cooperation mechanisms were introduced to account for special situations, diversify the solutions and create new chains.

In the next section we evaluate the performances of our method. We compare it to four other local-search and population-based metaheuristics on unconstrained mono-objective optimization problems.

4 Experiments and Results

This section compares the performances of CoBOpti with four other methods cited in section 2: Simulated Annealing (SA), Generalized Simulated Annealing (GSA), Differential Evolution (DE) and Particle Swarm Optimization (PSO).

Section 4.1 presents the different test functions used to test the performances; section 4.2 describes the protocole for comparing the performances of CoBOpti

with other selected methods; section 4.3 presents the results of the experiments; finally, results are discussed in section 5.

4.1 Test Functions

For the performance comparison experiments, four functions have been selected: Gramacy and Lee (domain: $[0.5, 2.5]$), Ackley (parameters: $d = 1$, $a = 20$, $b = 0.2$, $c = 2\pi$; domain: $[-32, 32]$), Rastrigin (parameter: $d = 1$; domain: $[-5.12, 5.12]$) and Levy function (parameter: $d = 1$; domain: $[-10, 10]$). These functions have been chosen because they feature many local minima, a single global minimum, and a single parameter [1,7,10,13].

4.2 Methods Comparison

The performances of each approach (SA, GSA, DE and PSO) are compared against CoBOpti's. They were all implemented in Python 3.8. GSA and DE were implemented using the `scipy.optimize.dual_annealing` and `scipy.optimize.differential_evolution` functions, PSO was implemented with `pyswarm.pso` package, and SA was a custom implementation. For GSA, DE and PSO, all optional parameters excepts those related to bounds, initial state and maximum number of iterations were let to their default value.

Control variables of CoBOpti are set as follows: $\varepsilon_{dist} = 10^{-4}$ (local minimum detection threshold), $\varepsilon_{same} = 0.01$ (minimum distance between local minima), $\delta_{min} = 10^{-4}$ (minimum step size during hill climbing phase), and $\varepsilon_{obj} = 5 \cdot 10^{-3}$ (precision threshold for global minimum objective value).

For every method, except PSO, the initial value v_{init} for each decision variable in a single run is selected by a Sobol Sequence. As values generated by this sequence are all in the $[0, 1]$ interval, they are adjusted to the variable's domain using the formula $v_{init} = s \cdot (d_{max} - d_{min}) + d_{min}$ where s is a value generated by the sequence. We did not specify v_{init} values for PSO as the implementation we used did not allow it.

Three metrics are defined: **success rate**, i.e. the ratio of executions that found the global minimum, **number of iterations**, **number of evaluations of the objective function**.

4.3 Results

Table 2 shows the success rate, mean number of iterations and function evaluations over 200 executions for each method and function, with a maximum of 1000 iterations.

CoBOpti was able to find the global minimum for all four functions. It took on average between 35 and 100 iterations to find the global minimum with a similar number of objective function evaluations.

The constant 1000 iterations for SA and GSA are explained by their stopping criterion. These methods rely on the number of elapsed iterations to compute

probability distributions: the more iterations have passed, the less likely the algorithm is to select a non-improving move. Once the allowed number of iterations has passed, no more non-improving moves can be selected and the algorithm stops. The visited point with the lowest objective value is then returned.

SA did not yield good results, except for Gramacy and Lee's function with nearly 100 % of success rate. It yielded very poor results for Ackley function with only 2 %. These results are coherent with what was described in the review (section 2).

GSA yielded very good results with 100 % on all functions. The number of objective function evaluations was two times higher than SA, around 2000.

DE's success rate is a bit lower than other methods except for SA. However, the mean number of iteration is quite low, staying between 8 and 50.

PSO was able to find the global minimum in all four cases with a low mean number of iterations, between 20 and 50. However, the mean number of function evaluations is higher than other methods, ranging from 2000 to more than 4500.

Table 2. Success rates, average number of iterations and objective function evaluations of tested methods

Method	**Function**	**Success rate**	**# of iterations**	**# of evaluations**
CoBOpti	G. & L.	100 %	49.31	50.31
	Ackley	100 %	95.94	96.94
	Rastrigin	100 %	80.69	81.69
	Levy	100 %	35.3	36.3
SA	G. & L.	99.5 %	1000	1000
	Ackley	2 %	1000	1000
	Rastrigin	10.5 %	1000	1000
	Levy	30 %	1000	1000
GSA	G. & L.	100 %	1000	2035.58
	Ackley	100 %	1000	2124.43
	Rastrigin	100 %	1000	2039.97
	Levy	100 %	1000	2019.60
DE	G. & L.	97.5 %	8.71	154.54
	Ackley	100 %	49.62	801.63
	Rastrigin	94 %	30.91	481.06
	Levy	100 %	50.45	773.75
PSO	G. & L.	100 %	20.61	2008.70
	Ackley	100 %	46.92	4638.51
	Rastrigin	100 %	25.57	2505.57
	Levy	100 %	20.02	1951.32

5 Analysis and Discussion

The initial assumption of continuity in function dynamics has been validated by the experiments on several standard functions. CoBOpti showed better success

rates than SA and DE, and nearly as good as GSA and PSO. Although the number of iterations of CoBOpti is comparable to that of DE and PSO, its number of function evaluations is several orders of magnitude lower.

This low number of objective function evaluations can be attributed to the fact that the objective function is evaluated only once per visited point. This behavior stems from the initial assumption that states that the dynamics of the objective function does not change significantly between two close points.

Execution times were not shown as differences between methods were not significant. This is most likely due to the relatively low complexity of the selected functions.

A sensitivity analysis should be done to test the influence of k_{dist} and k_{prop} on CoBOpti's performances.

CoBOpti was only tested on mono-objective optimization problems with a single decision variable. Further research is needed to generalize this approach to multi-objective global optimization problems with multiple decision variables. The core principle should stay similar to what was presented in this paper. New cooperation mechanisms should be added to select which objective to minimize and which decision variables to tune at each cycle.

Other experiments could be conducted with other complex functions. As real-world applications are subject to noisy data, resilience to such noise has to be tested.

6 Conclusion

In this paper, CoBOpti, a new metaheuristic for global optimization, was presented. It is based on a hypothesis of local continuity of the dynamics of the objective function. CoBOpti explores the search space by relying on the cooperation of visited solutions based on this hypothesis.

This paper focuses on mono-objective global optimization problems with a single decision variable. Experiments showed that CoBOpti needs less objective function evaluations than other common metaheurstic methods while maintaining similar or better success rates on 1D-functions.

CoBOpti is a promising proposition for use in on-line calibration. Indeed, its low number of objective function evaluations would be useful in the context of on-line calibration of complex simulation models with computationally intensive objective functions. This property could help reduce the time required to calibrate these kinds of models.

References

1. Alauddin, M.: Mosquito flying optimization (MFO). In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). pp. 79–84 (Mar 2016). https://doi.org/10.1109/ICEEOT.2016.7754783
2. Arsenault, R., Poulin, A., Côté, P., Brissette, F.: Comparison of Stochastic Optimization Algorithms in Hydrological Model Calibration. Journal of

Hydrologic Engineering **19**(7), 1374–1384 (Jul 2014). https://doi.org/10.1061/(ASCE)HE.1943-5584.0000938, https://ascelibrary.org/doi/abs/10.1061/(ASCE)HE.1943-5584.0000938, publisher: American Society of Civil Engineers
3. Cho, J.H., Wang, Y., Chen, I.R., Chan, K.S., Swami, A.: A Survey on Modeling and Optimizing Multi-Objective Systems. IEEE Communications Surveys Tutorials **19**(3), 1867–1901 (2017). https://doi.org/10.1109/COMST.2017.2698366, conference Name: IEEE Communications Surveys Tutorials
4. Fan, C., Zhang, C., Yahja, A., Mostafavi, A.: Disaster City Digital Twin: A vision for integrating artificial and human intelligence for disaster management. International Journal of Information Management **56**, 102049 (Feb 2021). https://doi.org/10.1016/j.ijinfomgt.2019.102049, https://www.sciencedirect.com/science/article/pii/S0268401219302956
5. Gendreau, M., Potvin, J.Y.: Tabu Search. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, pp. 165–186. Springer US, Boston, MA (2005). https://doi.org/10.1007/0-387-28356-0_6, https://doi.org/10.1007/0-387-28356-0_6
6. Gendreau, M., Potvin, J.Y. (eds.): Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 146. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-1-4419-1665-5, http://link.springer.com/10.1007/978-1-4419-1665-5
7. Gramacy, R.B., Lee, H.K.H.: Cases for the nugget in modeling computer experiments. Statistics and Computing **22**(3), 713–722 (May 2012). https://doi.org/10.1007/s11222-010-9224-x, https://doi.org/10.1007/s11222-010-9224-x
8. Ma, J., Dong, H., Zhang, H.M.: Calibration of Microsimulation with Heuristic Optimization Methods. Transportation Research Record **1999**(1), 208–217 (Jan 2007). https://doi.org/10.3141/1999-22, https://doi.org/10.3141/1999-22, publisher: SAGE Publications Inc
9. Opara, K.R., Arabas, J.: Differential Evolution: A survey of theoretical analyses. Swarm and Evolutionary Computation **44**, 546–558 (Feb 2019). https://doi.org/10.1016/j.swevo.2018.06.010, https://www.sciencedirect.com/science/article/pii/S2210650217304224
10. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Schwefel, H.P., Männer, R. (eds.) Parallel Problem Solving from Nature — PPSN III. pp. 249–257. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_269
11. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization **11**(4), 341–359 (Dec 1997). https://doi.org/10.1023/A:1008202821328, https://doi.org/10.1023/A:1008202821328
12. Sörensen, K., Sevaux, M., Glover, F.: A History of Metaheuristics. Handbook of Heuristics **to appear** (Jan 2017)
13. Valdez, F., Melin, P.: Parallel Evolutionary Computing using a cluster for Mathematical Function Optimization. In: NAFIPS 2007 - 2007 Annual Meeting of the North American Fuzzy Information Processing Society. pp. 598–603 (Jun 2007). https://doi.org/10.1109/NAFIPS.2007.383908
14. Zhang, Y., Wang, S., Ji, G.: A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Mathematical Problems in Engineering **2015**, e931256 (Oct 2015). https://doi.org/10.1155/2015/931256, https://www.hindawi.com/journals/mpe/2015/931256/, publisher: Hindawi