Evaluating AI-Generated Surrogate Optimization Algorithms for Expensive Black-Box Optimization

Rommel G. Regis(✉)

Saint Joseph's University, Philadelphia, PA 19131, USA
rregis@sju.edu

Abstract. Generative AI tools have become adept at writing code for various software development tasks. This paper evaluates the quality of the algorithms and Matlab codes for surrogate optimization that were generated by ChatGPT 5.2. The AI was prompted to write Matlab codes that are suitable for computationally expensive black-box optimization, particularly Bayesian optimization and radial basis function (RBF) methods. The AI-generated Matlab codes for surrogate optimization were compared with an original human-generated Matlab implementation of an established RBF method called Multistart Local Metric Stochastic RBF (MLMSRBF) on eight widely used test problems with up to 32 dimensions and on a 36-dimensional environmental engineering application. The AI-generated Matlab codes are well-structured and they implement and combine well-known surrogate optimization approaches. Among the codes it generated was a somewhat novel RBF algorithm that uses a strategy commonly associated with Bayesian optimization. Moreover, the performances of the AI-generated TuRBO-style Bayesian optimization code and the AI-generated Stochastic RBF-style code are comparable with the performance of the MLMSRBF code on the test problems. This suggests that AI tools can produce high quality surrogate optimization codes and have great potential to assist in extending or developing novel algorithms for expensive black-box optimization.

Keywords: Surrogate optimization · black-box optimization · generative AI · Bayesian optimization · radial basis functions

1 Introduction

Generative AI tools are increasingly being used to solve problems in Operations Research and Mathematical Optimization [23]. In particular, ChatGPT has been used to formulate problems in engineering optimization [21]. It has been used to develop solution approaches for variants of the Shortest Path Problem and to write Python codes that implement the algorithms [10]. ChatGPT has also been used to generate a C code that implements a swarm intelligence algorithm to

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16760, pp. 247–257, 2027.
https://doi.org/10.1007/978-3-032-30521-3_16

solve univariate global optimization problems [17]. This paper considers ChatGPT for generating codes that implement algorithms for expensive black-box optimization.

Due to the ubiquity of expensive simulation-based black-box optimization problems in many engineering applications, there is great demand for surrogate optimization algorithms. These methods use surrogate models that approximate the objective and/or constraint functions in every iteration and uses these surrogates to guide the selection of subsequent points for function evaluations. Popular methods include Bayesian optimization [2,6,7,9], which uses Gaussian Process (GP) surrogates [13], and radial basis function methods [8,15]. While there are some commercial and publicly available codes that implement surrogate methods, they are sometimes only available in one or two programming languages. For example, the TuRBO algorithm [5] that implements local Bayesian optimization is available in Python. The SACOBRA algorithm [3], which uses radial basis function (RBF) interpolation for constrained black-box optimization, is available in R. The Stochastic RBF algorithm was written in Matlab and its variants are also available in Python through the Surrogate Optimization Toolbox (pySOT) [4]. Moreover, commercial packages such as BayesOpt and surrogateopt require Matlab's Statistics and Machine Learning Toolbox [20] and the Global Optimization Toolbox [19], respectively. Potential users who need one of these algorithms might find that it is not available in their preferred language and could resort to AI tools to code the algorithm. Additionally, other users who need to solve expensive black-box optimization problems but are not be familiar with the literature might prompt an AI tool to generate such a method. As generative AI tools become more adept at writing codes, it would be interesting to explore how well AI tools can generate codes that implement algorithms for expensive black-box optimization, particularly surrogate optimization methods.

This paper explores the capability of ChatGPT 5.2 to generate Matlab codes for solving expensive black-box optimization problems. When prompted for such a code, GPT-5.2 generated one that implements Bayesian optimization with the classic Expected Improvement criterion [9]. It was also prompted to generate alternative methods that use radial basis function (RBF) surrogates and a local variant of Bayesian optimization that is more suitable for higher dimensional problems. One of the AI-generated codes was a hybrid of ideas from Bayesian optimization and RBF methods producing a somewhat novel surrogate optimization algorithm. The AI-generated Matlab codes for surrogate optimization were compared with an original human-generated Matlab implementation of an established RBF method called Multistart Local Metric Stochastic RBF (MLMSRBF) [15] on widely used test problems. The Stochastic RBF method was chosen for comparison since the paper that introduced it was recently given the INFORMS Journal on Computing Test of Time Award for 2005–2009 [16]. The Matlab codes generated by GPT-5.2 are well-structured and well-documented and implement well-known surrogate optimization approaches. The AI-generated Stochastic RBF-style code and the AI-generated TuRBO-style Bayesian optimization code yielded comparable results to those obtained by the non-AI MLMSRBF

code on several test problems, including an environmental engineering application. This suggests that AI tools such as GPT-5.2 can produce codes for surrogate optimization with comparable performance to a human-generated code.

2 Surrogate Optimization Methods

In this study, we evaluate the AI-generated Matlab codes for bound-constrained expensive black-box optimization problems, which have the form:

$$\begin{array}{c} \min\ f(x) \text{ s.t.} \\ \ell \le x \le u \\ x \in \mathbb{R}^d \end{array} \tag{1}$$

where $\ell, u \in \mathbb{R}^d$ and the objective functions f is black-box and computationally expensive. A similar experiment can be done for optimization problems with constraints, but this will be the subject of future work.

Surrogate optimization methods for problem (1) typically follow the same structure (Algorithm 1). Below, n_0 is the number of initial sample points, n_{max} is the maximum number of function evaluations, $\mathcal{P}_n = \langle x_1, x_2, \ldots, x_n \rangle$ denotes the ordered set of the first n sample points, and $f(\mathcal{P}_n) = \langle f(x_1), f(x_2), \ldots, f(x_n) \rangle$ is the corresponding ordered set of function values.

Algorithm 1 Surrogate Optimization Framework

Inputs: $n_0 \ge d+1$, $n_{max} > n_0$
Choose n_0 sample points $\mathcal{P}_{n_0} = \langle x_1, \ldots, x_{n_0} \rangle \subset [\ell, u]$ ▷ space-filling design
Evaluate f at the points in $\mathcal{P}_{n_0}$ and set $n \leftarrow n_0$
while $n < n_{max}$ **do**
  Fit surrogate model $s_n(x)$ using data from $\mathcal{P}_n = \langle x_1, \ldots, x_n \rangle$ and $f(\mathcal{P}_n)$
  Select $x_{n+1} \in [\ell, u]$ using some strategy and using info from $\mathcal{P}_n$, $f(\mathcal{P}_n)$ and $s_n(x)$
  Evaluate $f(x_{n+1})$ and set $n \leftarrow n+1$
end while
Return $x_n^* \leftarrow \text{argmin}_{\widetilde{x} \in S_n} f(\widetilde{x})$

In Algorithm 1, the initial sample points $\mathcal{P}_{n_0}$ normally come from a space-filling design such as a Latin hypercube design (LHD). The surrogate $s_n(x)$ is an approximation of $f(x)$ and it may be an interpolating model (i.e., $s_n(x_i) = f(x_i)$ for all $i = 1, \ldots, n$) or a non-interpolating model. Two of the most widely used surrogate models are Gaussian Process (GP) models [13] and Radial Basis Function (RBF) models [12], which are both interpolating models. Because evaluating f is expensive, we often carefully select the next sample point x_{n+1} using some scheme that is either expected (though not guaranteed) to yield a better function value or improve the quality of the current surrogate, or both. The strategy to select the next sample point x_{n+1} is called an *infill criterion* and this usually involves optimizing an *acquisition function*. Surrogate optimization methods are

typically distinguished by the type of surrogate for f and by the infill criterion or acquisition function used to select the next sample point x_{n+1}.

The most popular surrogate optimization approach at present is Bayesian optimization [2,7,9], which uses Gaussian Process surrogates [13] together with some widely used acquisition functions such as the expected improvement and lower confidence bound criteria. A tutorial on this method can be found in [6].

An alternative to Bayesian optimization includes RBF methods such as the RBF method by Gutmann [8] and the Stochastic RBF method [15]. RBF methods, particularly variants of the latter, have been quite successful on high-dimensional test problems and application problems involving hundreds of decision variables (e.g., [14]).

3 Using ChatGPT to Generate Matlab Codes for Expensive Black-Box Optimization

Below, we present some prompts used to generate Matlab codes for expensive black-box optimization. A total of four Matlab codes were generated by the AI, two implement Bayesian optimization while the other two implement RBF methods. In all cases, the codes produced were well-structured and they include proper documentation and references, which are mostly accurate.

First, the following prompt was used on GPT-5.2 to generate a Matlab code that is suitable for expensive black-box global optimization where the variables are subject only to bound constraints.

Please provide a Matlab function that searches for a global minimum point of a computationally expensive black-box objective function subject only to bound constraints. Use an algorithm that attempts to find the global minimum using only a relatively small number of objective function evaluations and without using the gradient of the objective function. The inputs to the Matlab function include the objective function to minimize, the bounds on the decision variables, and the computational budget in terms of number of function evaluations. If using algorithms and ideas in the literature, please provide proper citations of books or articles. Please also demonstrate how to use this Matlab function on three very different widely used test functions whose dimensions are 4, 7 and 10.

The AI generated a Matlab code called bbopt_global_min.m that implements standard Bayesian optimization with the Expected Improvement (EI) criterion. This outcome was expected since Bayesian optimization is the currently the most popular algorithm for expensive black-box optimization. Moreover, the EI criterion is the classic acquisition function associated with Bayesian optimization that was introduced in the paper on Efficient Global Optimization (EGO) [9]. However, unlike the original implementation of EGO and other Bayesian optimization codes, the AI-generated code simply maximizes the EI criterion over random candidate points instead of doing a more thorough search. This idea of optimizing an acquisition function over random candidate points is similar to that used by the Stochastic RBF method [15] and it results in much faster

code, especially when using a small number of random candidate points relative to the dimension of the problem. In this case, the AI-generated code uses a default of 5000 candidate points.

Next, GPT-5.2 was given the following prompt to generate an alternative surrogate optimization method that uses RBF surrogates:

Are you able to provide a similar Matlab function for computationally expensive bound-constrained black-box global optimization that uses radial basis function surrogates instead of Gaussian process surrogates with the Bayesian optimization idea? Again, please provide proper citations of books or articles.

The AI then generated a code called rbfopt_global_min.m that uses RBF interpolation and ideas from the Stochastic RBF paper [15] such as the use of the distance-based criterion. However, the acquisition function it used was not the same as that in the original paper. Instead, the structure is more similar to that of the Lower Confidence Bound idea that is also used in Bayesian optimization except that the uncertainty is replace by a distance-based proxy where points that are farther away from current sample points are associated with higher uncertainty in the surrogate approximation. Hence, GPT-5.2 seems to have generated a novel and reasonable surrogate optimization method that is a hybrid of ideas used in RBF methods and Bayesian optimization.

At the end of the response, the AI also offered to generate a code that is closer to the Stochastic RBF algorithm by Regis and Shoemaker [15]. Hence, the following prompt was then given to the AI:

Please provide a closer-to-Regis/Shoemaker variant of the code that uses RBF with the same input-output structure.

This resulted in another code called rbfopt_global_min_srbf.m that was indeed more similar to the Stochastic RBF method called MLMSRBF [15] except that the default weights for the RBF and distance criteria in the acquisition function were different from the one used in the original MLMSRBF algorithm. Instead, they were taken from an extension of the algorithm to constrained black-box optimization [14]. To come up with an even closer implementation of MLMSRBF, the AI-generated code was also run with the same weights used as in the original MLMSRBF algorithm. This gave rise to two variants of this code: RBFOPT-SRBF-A (uses the default weights provided by the AI) and RBFOPT-SRBF-B (uses the same weights as MLMSRBF).

Finally, the BBOPT algorithm that uses Bayesian optimization yielded relatively poor results on the higher dimensional test problems, so the AI was prompted to generate a variant of Bayesian optimization that runs more quickly and that is more suitable for higher dimensional problems:

Can you create a variant of the bbopt_global_min.m code that runs more quickly and that is more suitable for higher dimensional optimization problems with 10 or more decision variables?

GPT-5.2 then generated the code bbopt_global_min_fast.m, which is a Matlab implementation of the TuRBO (Trust Region Bayesian Optimization) algorithm that was designed for higher dimensional problems. The original TuRBO algorithm is implemented in Python [5], so the AI-generated code is

convenient for users whose simulation code for the black-box function runs on Matlab. TuRBO employs a trust region approach with a Gaussian Process model built using the k nearest sampled points from the trust region center. The next sample point is obtained by minimizing a lower confidence bound (LCB) acquisition function within the current trust region. This approach is meant to promote local search, which facilitates progress on high dimensional problems.

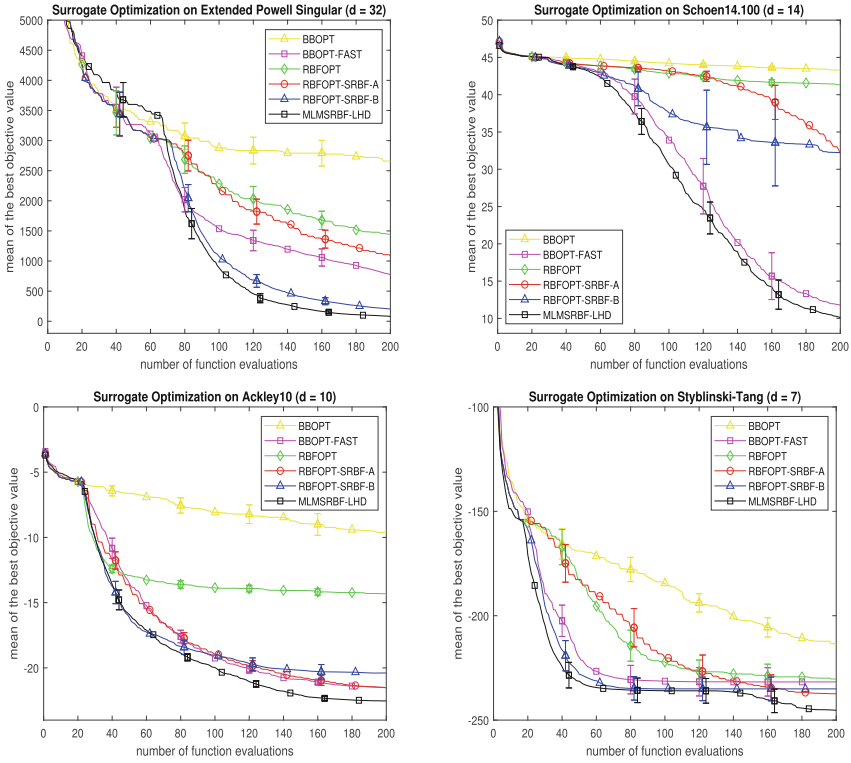

Fig. 1. Mean of the best objective function values (over 30 trials) obtained by the AI-generated surrogate optimization codes and the MLMSRBF-LHD algorithm on the Extended Powell Singular (top left), Schoen14.100 (top right), Ackley10 (bottom left) and Styblinski-Tang (bottom right) test problems. In all plots, the error bars correspond to 95% t confidence intervals for the mean.

4 Experimental Setup

We compare Matlab codes generated by GPT-5.2 with an original human-generated Matlab code of the Multistart Local Metric Stochastic RBF (MLMSRBF) [15] on eight well-known test problems and on a 36-dimensional groundwater bioremediation application described in [22]. Groundwater bioremediation

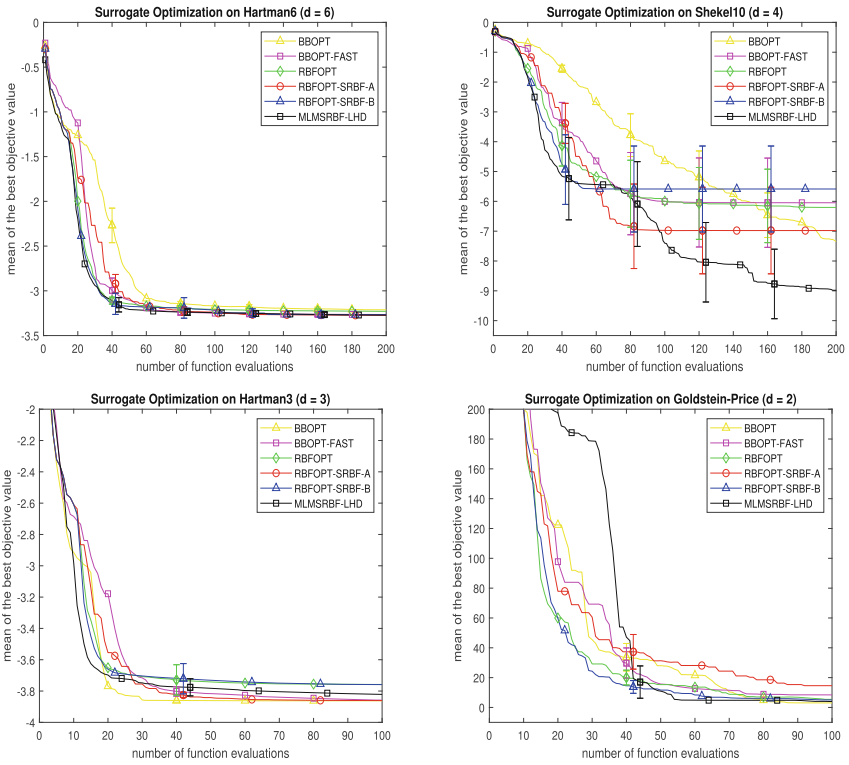

Fig. 2. Mean of the best objective function values (over 30 trials) obtained by the AI-generated surrogate optimization codes and the MLMSRBF-LHD algorithm on the Hartman6 (top left), Shekel10 (top right), Hartman3 (bottom left), and Goldstein-Price (bottom right) test problems. In all plots, the error bars correspond to 95% t confidence intervals for the mean.

is the process of promoting the growth of microorganisms that are then used to break down hazardous contaminants in the groundwater. Each of the AI-generated codes BBOPT, BBOPT-FAST, RBFOPT were run using the default settings. The AI-generated code RBFOPT-SRBF, which is an AI implementation that resembles MLMSRBF was run with two variants: RBFOPT-SRBF-A uses the default settings used by the AI while RBFOPT-SRBF-B is the exactly the same except it uses the same weights for the RBF and distance criteria as in MLMSRBF. The human-generated MLMSRBF code that was run comes from the GitHub repository https://github.com/INFORMSJoC/1060.0182 and used mostly the default settings in [15] except it used an approximate maximin Latin hypercube design (LHD) of size $2(d+1)$ instead of a symmetric LHD. The test problems and their dimensions are the Extended Powell Singular ($d = 32$) [11], Ackley10 ($d = 10$) [1], Styblinski-Tang ($d = 7$) [18] and five of the test prob-

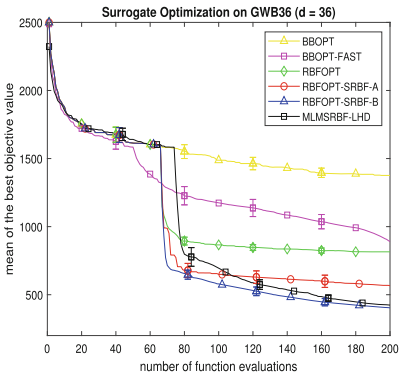

Fig. 3. Mean of the best objective function values (over 30 trials) obtained by the AI-generated surrogate optimization codes and the MLMSRBF-LHD algorithm on the 36-dimensional groundwater bioremediation application GWB36. In both plots, the error bars correspond to 95% t confidence intervals for the mean.

lems used in [15], namely, Schoen14.100 ($d = 14$), Hartman6 ($d = 6$), Shekel10 ($d = 4$), Hartman3 ($d = 3$) and Goldstein-Price ($d = 2$).

To obtain average case performance, each of the AI-generated codes and the original MLMSRBF code were run for 30 trials on each of the test problems and on the groundwater bioremediation application.

5 Results and Discussion

Figures 1-2 show the plots of the mean best objective function value (over 30 trials) obtained by the AI-generated surrogate optimization codes and the MLMSRBF-LHD algorithm on the eight synthetic test problems. The error bars correspond to 95% t confidence intervals for the mean. Moreover, Fig. 3 show the corresponding plots on the 36-D groundwater bioremediation application.

The plots show that the original MLMSRBF-LHD algorithm generally performs better than the AI-generated surrogate optimization codes on five of the synthetic test problems. Moreover, its performance is comparable with that of the AI-generated surrogate optimization codes on Hartman6 and Hartman3. However, the RBFOPT-SRBF-B code, which GPT-5.2 designed to be more similar to the MLMSRBF-LHD algorithm, was generally better than the latter on the 36-D groundwater bioremediation application and on the 2-D Goldstein-Price problem. Moreover, the performance of RBFOPT-SRBF-B is close to that of MLMSRBF-LHD on the 32-D Extended Powell Singular, the 7-D Styblinski-Tang, Hartman6 and Hartman3. The BBOPT-FAST code has comparable performance to MLMSRBF-LHD on the Goldstein-Price, Hartman3 and Hartman6, and its mean objective function value is not too far from that of the MLMSRBF-LHD on Schoen14.100, Ackley10, and Styblinski-Tang. Overall, the performances of RBFOPT-SRBF-B and BBOPT-FAST are close to that of MLMSRBF-LHD

on the test problems. Additionally, the somewhat novel RBF-OPT performs better than BBOPT-FAST on GWB36 and Goldstein-Price and has comparable performance to BBOPT-FAST on Hartman6, Shekel10 and Hartman3.

6 Summary and Future Work

This study compared the Matlab codes for surrogate optimization that were generated by ChatGPT 5.2 with a human-generated original Matlab code that implements Multistart Local Metric Stochastic RBF (MLMSRBF), which is an established RBF method for expensive black-box optimization. The codes were compared on eight well-known optimization test problems with up to 32 dimensions and on a 36-dimensional groundwater bioremediation application. The codes generated by GPT-5.2 are well-organized and well-documented and they implement standard surrogate optimization approaches. In the case of the AI-generated RBFOPT code, GPT-5.2 created a somewhat novel RBF algorithm that minimizes a lower confidence bound acquisition function that is commonly used in Bayesian optimization. The numerical results show that the AI-generated BBOPT-FAST and RBFOPT-SRBF-B codes achieved comparable performance to the human-generated MLMSRBF code on the test problems. Moreover, on the groundwater application, RBFOPT-SRBF-B performed better than the human-generated MLMSRBF code while RBFOPT-SRBF-A achieved comparable performance with MLMSRBF. BBOPT-FAST is an AI-generated TuRBO-style Bayesian optimization code while RBFOPT-SRBF-B is an AI-generated Stochastic RBF-style code that uses weights for the RBF and distance criteria that are similar to MLMSRBF. These results indicate that AI tools can be used to generate high quality Matlab codes for surrogate-based expensive black-box optimization. They also suggest great potential for AI tools to assist in developing novel optimization algorithms by combining ideas from multiple methods. Future work will experiment with using the AI to generate promising infill criteria for expensive black-box optimization. We also examine AI-generated Matlab codes for constrained optimization and multi-objective optimization and codes in other languages such as R or Python.

Acknowledgments. This study used ChatGPT Enterprise with GPT-5.2 from the Wharton School of the University of Pennsylvania.

Disclosure of Interests. The author has no competing interests to declare that are relevant to the content of this article.

References

1. Ackley, D.H.: A Connectionist Machine for Genetic Hillclimbing. Kluwer Academic Publishers, Boston, MA (1987)
2. Archetti, F., Candelieri, A.: Bayesian optimization and data science (Vol. 849). Springer, Cham (2019)

3. Bagheri, S., Konen, W., Emmerich, M., Bäck, T.: Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. Appl. Soft Comput. **61**, 377–393 (2017)
4. Eriksson, D., Bindel, D., Shoemaker, C.A.: pySOT and POAP: An event-driven asynchronous framework for surrogate optimization. arXiv preprint arXiv:1908.00420 (2019)
5. Eriksson, D., Pearce, M., Gardner, J., Turner, R.D., Poloczek, M.: Scalable global optimization via local Bayesian optimization. Adv. Neural Inform. Process. Syst. **32** (2019)
6. Frazier, P.I.: A tutorial on Bayesian optimization. arXiv preprint arXiv:1807.02811 (2018)
7. Garnett, R.: Bayesian optimization. Cambridge University Press (2023)
8. Gutmann, H.-M.: A radial basis function method for global optimization. J. Global Optim. **19**(3), 201–227 (2001)
9. Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. J. Global Optim. **13**(4), 455–492 (1998)
10. Luzzi, M., Guerriero, F., Maratea, M., Greco, G., Garofalo, M.: Chatgpt and operations research: evaluation on the shortest path problem. Soft Comput. **29**, 1407–1418 (2025). https://doi.org/10.1007/s00500-025-10505-2
11. Moré, J., Garbow, B., Hillstrom, K.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**, 17–41 (1981)
12. Powell, M.J.D.: The theory of radial basis function approximation in 1990. In: Light, W. (ed.) Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions, pp. 105–210. Oxford University Press, Oxford, U.K. (1992)
13. Rasmussen, C. E.: Gaussian Processes for Machine Learning. The MIT Press (2006)
14. Regis, R.G.: Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. Comput. Oper. Res. **38**(5), 837–853 (2011)
15. Regis, R.G., Shoemaker, C.A.: A stochastic radial basis function method for the global optimization of expensive functions. INFORMS J. Comput. **19**(4), 497–509 (2007)
16. Smith, A.E.: Note from the Editor. INFORMS J. Comput. **37**(4), iv–viii (2005). https://doi.org/10.1287/ijoc.2025.ed.v37.n4
17. Sorokovikov, P., Gornov, A., Zarodnyuk, T.: Numerical Investigation of the Swarm Intelligence Algorithm Obtained Using ChatGPT for Univariate Global Optimization. In: Eremeev, A., Khachay, M., Kochetov, Y., Mazalov, V., Pardalos, P. (eds) Mathematical Optimization Theory and Operations Research: Recent Trends. MOTOR 2024. Communications in Computer and Information Science, vol 2239. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-73365-9_25 (2024)
18. Styblinski, M.A., Tang, T.: Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing. Neural Netw. **3**(4), 467–483 (1990)
19. The MathWorks Inc.: Global Optimization Toolbox, Version R2025b. The MathWorks Inc., Natick, Massachusetts, United States (2025)
20. The MathWorks Inc.: Statistics and Machine Learning Toolbox, Version R2025b. The MathWorks Inc., Natick, Massachusetts, United States (2025)
21. Vu, N.G.H., Wang, K., Wang, G.G.: Effective prompting with ChatGPT for problem formulation in engineering optimization. Eng. Optim. **57**(12), 3677–3694 (2025)

22. Yoon, J.-H., Shoemaker, C.A.: Comparison of optimization methods for groundwater bioremediation. J. Water Resour. Plann. Manage. **125**, 54–63 (1999)
23. Zhou, Q., Sheu, J. B.: The use of Generative Artificial Intelligence (GenAI) in operations research: review and future research agenda. J. Oper. Res. Society 1–21. https://doi.org/10.1080/01605682.2025.2561762 (2025)