Optimization of Artificial Landscapes with a Hybridized Firefly Algorithm

Kevin Saner, Kyle Smith, Thomas Hanne, and Rolf Dornberger
School of Business, University of Applied Sciences and Arts Northwestern Switzerland, Olten/Basel, Switzerland
Email: {kevin.saner, kyle.smith}@students.fhnw.ch, {thomas.hanne, rolf.dornberger}@fhwn.ch

***Abstract*—This paper shows how the metaheuristic Firefly Algorithm (FA) can be enhanced by hybridization with a genetic algorithm to achieve better results for optimization problems. The authors examine which configuration of the hybridized FA performs best during a number of computational tests. The performance of the hybrid FA is compared with that of the regular FA in solving test functions for single-objective optimization problems in two and n-dimensional spaces. The key findings are that more complex optimization problems benefit from the hybrid FA because it outperforms the basic FA. In addition, some useful parameters settings for the suggested algorithm are determined.**

***Index Terms*—metaheuristics, swarm intelligence, firefly algorithm, genetic algorithm, hybridization, single-objective optimization, artificial landscapes, performance evaluation**

I. Introduction

Metaheuristic algorithms can be defined as stochastic algorithms which use randomization and global exploration to find solutions for problem models dealing with nonlinear modeling and global optimization. They are a general algorithmic framework that uses approximation to address complex problems and can produce acceptable results through trial and error to solve a complex problem in a reasonable time. Metaheuristics solve problem models especially in situations in which it is impossible to search every possible solution or combination. Therefore, the objective is to find good feasible solutions in an acceptable time frame. For some classes of problems, this means that metaheuristics do not guarantee that a globally optimal solution is found. For instance, according to Gandomi *et al.* [1] there are no efficient algorithms for hard optimization problems, especially for NP-hard optimization problems. Therefore, the choice for using a specific metaheuristic algorithm depends on the problem model, the computational resources, and time constraints [1].

Two components are used in metaheuristic algorithms: intensification and diversification which are also denoted as exploitation and exploration [2]. Diversification describes how diverse solutions are generated to explore a search space on a global scale, while intensification describes the search of the algorithm in a local region by exploiting the information that a current good solution is found in that region. Diversification uses randomization to avoid that solutions get trapped at local optima. At the same time, diversification increases the diversity of the solutions found. By combining the two components intensification and diversification with the selection of the best solutions, it usually can be ensured that the global optimality is achieved [3].

Evolutionary Algorithms (EAs) are nature-inspired metaheuristics and act as stochastic global optimizers which mimic main aspects of biological evolution. They are population-based algorithms that learn from past searches by using a group of individuals or agents. The performance of an EA can be tested with so-called test functions for optimization (artificial landscapes) to discover their strengths and weaknesses. Many EAs share similarities and can therefore be combined. To mitigate possibly insufficient performance by an EA, researchers have developed methods to enhance the behavior of the algorithms - either to address premature convergence or to slow convergence, which ultimately may lead to better results. Hybridization is a trending and commonly used method [3]. In this paper the performance of such a hybrid algorithm based on the Firefly Algorithm (FA), a promising rather recent metaheuristic, combined with the Genetic Algorithm (GA) is tested in 2 dimensions and *n* dimensions on artificial landscapes and compared with the results of the standard FA.

Section II starts with an introduction to the identified gap of using hybrid metaheuristic algorithms for optimization problems. Section III introduces the FA as well as the GA. Section IV highlights which different artificial landscapes were chosen to test the performance of hybrid algorithm. In Section V, we summarize the findings and Section VI presents the conclusions.

II. Overview: Hybridization and Artificial Landscapes

A. Hybridization for Performance Improvement

Hybridization of EAs is a method of combining the advantages of metaheuristic algorithms to form a hybrid algorithm making use of mutual advantages of the considered methods. In 2012, Rodriguez *et al.* [4] identified that there were more publications on EAs hybridized with Simulated Annealing than on other hybridized EAs. Yang *et al.* [5] found in a literature review from 2015 that hybrid algorithms are being developed in

Manuscript received December 6, 2021; revised May 22, 2022.

doi: 10.12720/jait.13.4.374-380

many fields to achieve better performance. Moreover, Yang pointed out that the hybridization of new nature-inspired algorithms may lead to novel characteristics. To create an efficient hybrid algorithm, the hybridization method should be based on mathematical theory and insightful analysis and not be the result of an arbitrary EA hybridization. The FA is part of the Swarm Algorithms which are occasionally also considered as Evolutionary Algorithms, while the GA belong to the Evolutionary Algorithms as well [6]. Based on literature that highlights the potential of a hybrid FA combined with the GA [5], the authors have chosen to test the advantages of the performance of the hybrid FA on artificial landscapes.

B. *Artificial Landscapes*

Test functions for optimization, also called artificial landscapes, are used to evaluate the characteristics of an optimization algorithm making use of versatile objective functions. They can be used to evaluate optimization algorithms according to characteristics such as the convergence rate, the precision, robustness, and general performance [7]. Artificial landscapes are essentially optimization problems in the form of mathematical functions for constrained or unconstrained, single-objective or multi-objective optimization. The test functions use *d* in their notation as the problem dimension and are optimized with a set of best suitable parameter values designed to allow the algorithm to find the best solution. Often, the best solution is intentionally hidden in a variety of suboptimal solutions within the problem landscape, visible in plots as hills and valleys. They must include good global searchability to avoid being trapped, as these suboptimal solutions are local minima or local maxima. Optimization algorithms, including metaheuristic algorithms, may find the best solution, but this is not always guaranteed and will not be achieved as quickly as possible. The algorithms that have better global searchability are hard to trap in sub-optimal locations. Algorithms that perform well on a set of numerical optimization problems are considered effective methods for solving real-world problems [7], [8].

The authors test the performance of the hybrid FA algorithm on the Ackley and the Sphere functions in different dimensions (Fig. 1). The Ackley function is a non-convex function and has many local minima. It is widely used as a performance testing problem because it poses a risk for an optimization algorithm to be trapped in one of the local minima, which would result in an inefficient exploration of the search space. The Sphere function is a simple bowl-shaped function and has one local minimum. Its form is regarded as continuous, convex, and unimodal [9]. Both functions are presented in Fig. 1 in their two-dimensional form.

The authors decided to test the hybrid FA in an experimental setup in a two-dimensional search space. It should deliver some insight into which parameter settings look promising. The most promising setups should then also be applied in a multidimensional search space. This should validate the test results or highlight differences. Further, some experimental tests were conducted in an approach to further improve the algorithm.

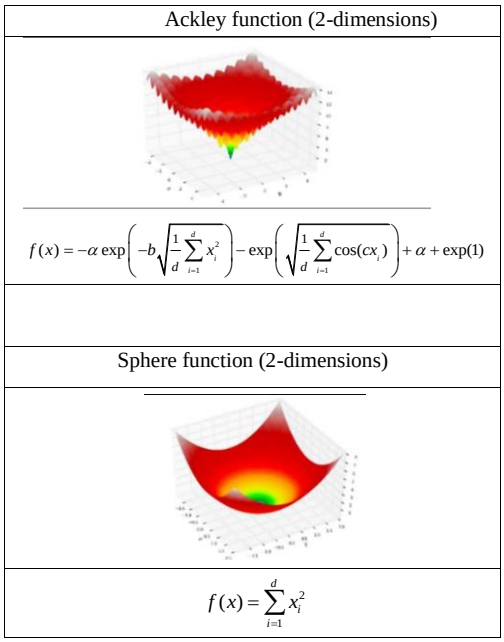

Figure 1. Artificial landscapes - The Ackley and the Sphere function in 2-dimensions.

III. The Firefly Algorithm and the Genetic Algorithm

A. *The Firefly Algorithm (FA)*

As part of the Swarm Intelligence Algorithms (SIA), the FA belongs to the group of nature-inspired metaheuristics which can be used to solve global optimum problems. The FA was inspired by the flashing behavior of fireflies. The attractiveness β of a firefly is directly proportional to its light intensity (brightness). Each firefly within the population moves towards another brighter firefly. Equation (1) below captures how the movement of a firefly *i* towards the position of another brighter and therefore more attractive firefly *j*. β_0 is the attractiveness at $r = 0$, α is a randomization parameter, and κ is defined as a random number taken from a uniform or Gaussian distribution. The values $r_{i,j}$ represent the distances between the firefly agents *i* and *j*. For each cycle of the FA, the positions of each pair of agents are updated [10], [11].

$$x_i(t+1) = x_i(t) + \beta_0 r^{-\gamma r_{i,j}^2}(x_j - x_i) + \alpha(\kappa - 0.5) \quad (1)$$

The FA is represented by the flow chart in Fig. 2.

The FA proved to be good at exploring the search space as well as for the exploitation of an attractive area of the search space. Its advantages are threefold: It can solve highly nonlinear, multi-modal optimization problems. No initial good solution is required to begin its iterations and it provides the adaptability to be combined with other advanced procedures [2]. During late iterations it was found that the basic FA is suffering from low exploration power [11]. This means it can become trapped at a local optimum. To mitigate such unwanted performance the enhancement method of hybridization can lead to better results.

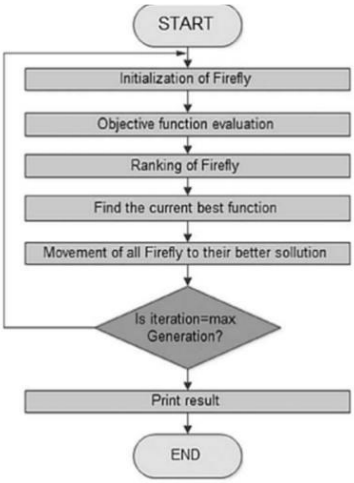

Figure 2. Flowchart of the FA.

B. The Genetic Algorithm (GA)

The GA is a well-known Evolutionary Algorithm created on the principles of genetics and natural selection. The GA is an iterative method which manipulates a population with a constant size. The agents in the population are called chromosomes and are evolved in a competitive procedure to search for the optimum solution. Each chromosome carries the encoding of a potential solution to solve the problem. The potential solution contains a set of elements called genes and can hold numerous values. For each iteration (generation), a new population of the same size is generated. The generation enters the procedure of applying the genetic operators: selection, crossover, and mutation. The procedure adapts the generation of chromosomes to the selection function and gradually the "better" chromosomes move towards the optimum [2].

Selection: During selection, the reproduction probability (p_i) for each individual agent is proportional to its fitness function value:

$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

Crossover: During the crossover procedure, a population is divided into two parts and each pair goes through the crossover process with a certain probability (p_c). It has been observed that the GAs strength depends on convergence and diversity which are strongly influenced by the crossover operator. There are different variants of the crossover found in the literature, such as the single-point crossover, two-point crossover and the arithmetic crossover.

Mutation: Finally, the agents in the population encounter the mutation process in which parts of the genes are randomly changed with a certain probability (p_m) [12].

Main advantages of the GA are its flexibility of dealing with many types of optimization problems. A GA's population can simultaneously use the search space in different areas because the various offspring solutions in the population behave as independent agents. Unlike most search methods, the GA only requires information about the quality of a solution specified by the fitness function. Various other optimization methods either require derivative information or other information about the optimization problem. In research, the GA was used frequently because it was found to be effective in exploiting the search space of promising areas by combining parts of good solutions [11], [13], [14]. A flowchart of the GA is presented in Fig. 3.

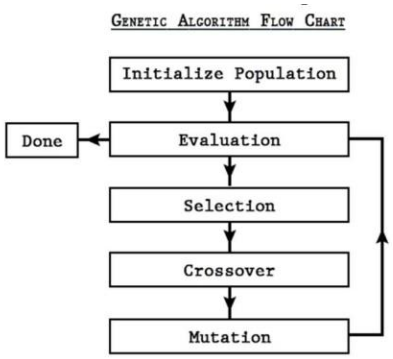

Figure 3. Flowchart of the genetic algorithm.

C. Categorization of Hybrid Algorithms

Yang *et al.* [5] stated that hybrid algorithms have the disadvantage of a missing naming convention in the current literature. A naming issue tends to arise when another algorithm is included into the main metaheuristic algorithm. Certain researchers choose to name their hybrid algorithm very differently, which results in confusing abbreviations. Therefore, a simpler taxonomy of hybrid algorithms has been suggested; hybrid algorithms can be categorized into two types, namely Collaborative Hybrids and Integrative Hybrids.

Collaborative Hybrids involve the combination of two or more algorithms that run in parallel or in a sequential manner and manipulate a population. A framework can use the first algorithm as a global optimizer and the second algorithm to perform the global search based on the unique capabilities of each algorithm.

Integrative Hybrids on the other hand have a master metaheuristic and the integrated algorithm is regarded as a subordinate. The master metaheuristic incorporates a partial procedure of the subordinate algorithm. Fig. 4 shows how a master metaheuristic integrates a specific procedure (GA's mutation procedure) from another algorithm to generate the new population [5].

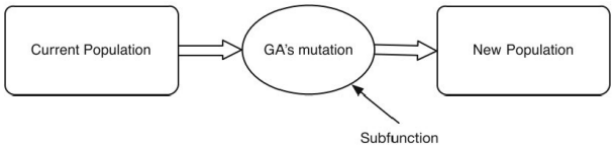

Figure 4. Integrative structure of a hybrid algorithm.

Research has found that if the FA is hybridized with a GA during later iterations, it would outperform the basic implementation of the FA. A hybrid solution is generated with a probability of 0.5 after 2/3 of the iterations [11], [13]. It is not evident how these parameters were determined.

We are looking forward to closing this research gap by conducting a set of computational experiments with the two above mentioned artificial landscapes. The chosen performance attributes for comparison are the optimization results. Thus, the described hybrid FA in this paper is an Integrative Hybrid, which uses the crossover operator from the GA and provides insights into the most suitable point within the hybrid FA to integrate the subordinate GA to achieve the best performance.

IV. Description of the Hybrid Optimization Method

Subsequently, it is explained how the master heuristic, the FA, and the subordinate GA are combined to work as an enhanced integrative hybrid FA.

A. Initialization

The optimization methods are based on the implementation for 2-dimensional and *n*-dimensional problems proposed by Yang [15]. The initial solutions are calculated randomly within the search space with uniform distribution.

B. Constraint Handling

The constraint handling in 2- and *n*-dimensional search space is done as follows. Assuming that there are bound constraints, a variable value x_i is set to the upper bound if it would surpass that value. It is set to the lower bound if it would fall below this value.

C. Hybridization

To improve the exploitation process, the basic implementation of the FA was complemented with the GA. With the help of the GA, a mutant solution is generated. Therefore, two new parameters were introduced.

The hybridization parameter *th* indicates after which number of iterations a mutant is generated. The hybridization parameter *th* is calculated as follows:

$$th = N - (N / d)$$

N is the number of iterations and d is an adjustable parameter.

The second parameter p_t represents the probability threshold. A modified offspring solution is generated when a uniformly distributed random number between 0 and 1 is higher than the applied threshold. So, if the threshold p_t is set to 0.5, there is 50% chance that a modified offspring is generated. The offspring (child 1) is created by applying a single-point crossover operator using the best solution and a random one (parents) from the population (Fig. 5). Child 1 now substitutes the worst solution in the population.

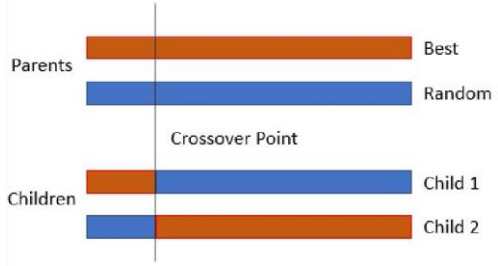

Figure 5. Single point crossover.

The crossover point(s) *cop* in the n-dimensional search space is defined as:

$$cop = rand(1, n)$$

A flowchart of the FA hybridized with the GA is shown in Fig. 6.

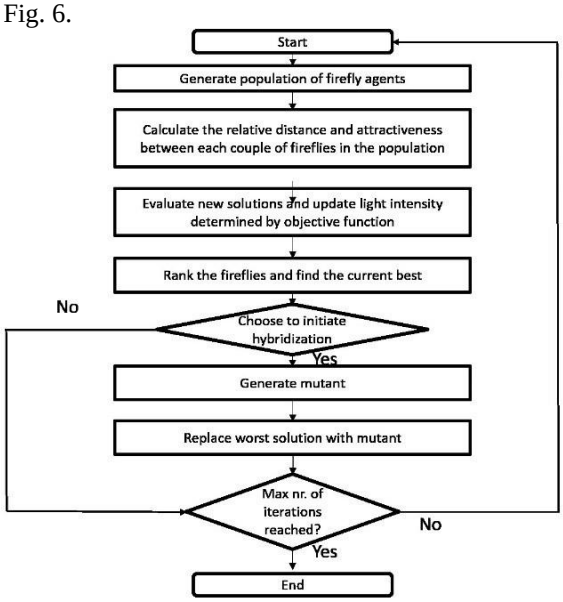

Figure 6. Hybrid FA flowchart.

V. Experiment Design

For the different experiments, the above-mentioned artificial landscapes are used. We firstly conducted the experiments on a 2-dimensional test function (Ackley) to gain insights into the improvement of a hybridized approach. The most promising setups would then be adapted to an n-dimensional test function (sphere) to validate the findings and gain more insights.

A. Parameter Settings

1) 2-Dimensional setup

In the 2-dimensional search space, the population size was set to *PS*=12. The number of iterations was set to *N*=100. Parameters specific to Yang's implementation of the FA for 2-dimensional problems are set as follows: initial value of randomness reduction parameter α=0.5, absorption coefficient γ=1, randomness reduction parameter δ=0.95 and attractiveness β_0=1 [15].

Parameters specific to the hybridized FA are set as follows: Divider d is set to 1…4 leading to a hybridization after *N*, 0.5**N*, 2/3**N* and 0.75**N* iterations, the probability threshold p_t of replacing the worst solution is set to 0. The crossover point is in the 2-dimensional setup always 1.

2) n-Dimensional setup

In the *n*-dimensional setup, the population size was set to *PS*=40. The number of iterations was set to *N*=800. The parameters specific to Yang's implementation of the FA for *n*-dimensional OPs are set as follows: initial value of randomness reduction parameter α=0.5, minimal value of attractiveness β=0.2 and absorption coefficient γ=1 [15].

The parameters specific to the hybridized implementation of the FA are set as follows: divider d=1…8, the probability threshold replacement p_t=0…0.5 with increments of 0.1. This results in probabilities that a modified offspring will be generated from 50% to 100%.

B. Performance Indicators

To test the performance of the 2-dimensional and the n-dimensional setup, all possible combinations of th and p_t are run 100 times. The performance of a setup was judged by the best, worst, and average of the objective functions result for each of these sets.

C. Test Results

All tests were performed on an Intel(R) Core i7-8665U processor @1.90GHz with 32GB RAM memory, Windows 10 Enterprise x64 operating system and MATLAB R2019b.

1) 2-Dimensional setup

Initially the experiment was designed to solve a 2-dimensional test function. The following Ackley function was used as the objective function.

$$f(x,y) = 20 * \exp\left[-0.2\sqrt{0.5(x^2+y^2)}\right] - \exp\left[0.5(\cos(2\pi x)+\cos(2\pi y))\right] + e + 20$$

The lower bound was set to -5 and the upper bound was set to 5. First, the standard FA was used to solve the test function. Then, the hybridized FA was used, so that the performance could be compared. The hybridization was therefore employed at different stages of the algorithm's runs. Every setting for th was analyzed during 100 runs of the algorithm. Comparing the results to the standard FA showed that no performance improvements could be achieved. If the standard FA was not trapped in a local optimum, it was in most cases also able to find the optimal solution. If the FA was trapped in a local optimum, a hybridized approach could not counteract, due to the previously stated fact, that exploitation is improved by hybridizing but not exploration. Thus, it was not possible to draw conclusions for further tests from the 2-dimensional setup.

2) n-Dimensional setup

As the 2-dimensional setup did not indicate which setups are promising, all combinations of divider and probability were tested for performance. As objective function the following sphere function was used:

$$f(x) = \sum_{i=1}^{b} (x_i - 1)^2$$

The sphere function was set to be 15-dimensional. The lower bound was set to 0 and the upper bound was set to 2 (note that the respective function shown in Fig. 1 is with values multiplied by -1). For comparison, the basic FA was first run with no hybridization. Then the different hybrid setups were run.

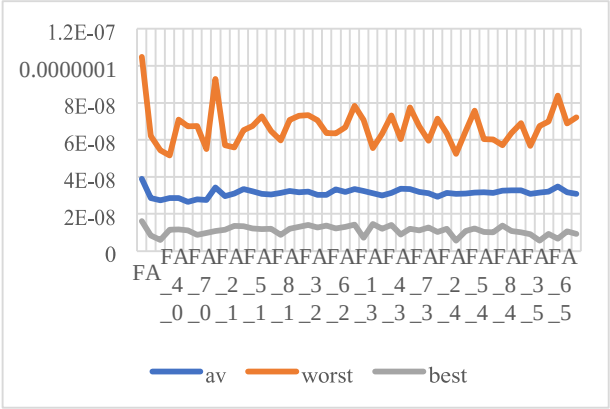

Figure 7. FA vs. hybrid FA.

Fig. 7 shows that the hybrid FA performs consistently better than the basic FA regarding the average outcome, except for the setup d=1 and p_t=0. The setup performs even worse than the basic FA and is not part of the figure. The figure shows that the best results can be attained when divider d is set to 6…8 and the probability threshold p_t to 0.

TABLE I. RESULTS OF OBJECTIVE FUNCTION FOR DIFFERENT SETUPS

Performance Data			
FA Setup	Average	Worst	Best
Basic FA	3.90315E-08	1.04747E-07	1.61495E-08
Hybrid FA (d=6, p_t=0)	2.66E-08	6.73509E-08	1.10824E-08
Hybrid FA (d=7, p_t=0)	2.78E-08	6.74577E-08	8.69141E-09
Hybrid FA (d=8, p_t=0)	2.76E-08	5.51E-08	9.91733E-09

Table I shows the results of the best runs. These setups not only perform best on average but considering at the worst runs, they are also less likely to get trapped in a local optimum.

The setup d=3 and p_t=0 also performed quite well, but to enable hybridization in late iterations seemed to be the more promising approach. So, this setup was not further investigated [11], [13].

Based on the above findings we conducted further experiments using a two-point crossover operator to generate the offspring. For these experiments only the most promising parameter setups were used.

TABLE II. 2-PCO (TWO-POINT CROSSOVER) FA COMPARED TO BASIC FA

Performance Data			
FA Setup	Average	Worst	Best
Basic FA	3.90315E-08	1.04747E-07	1.61495E-08
2 PCO FA (d=6, p_t=0)	2.72749E-08	5.76662E-08	9.32E-09
2 PCO FA (d=7, p_t=0)	2.98123E-08	7.24957E-08	1.16E-08
2 PCO FA (d=8, p_t=0)	3.05E-08	6.54634E-08	1.18563E-08

Table II shows that the FA algorithm hybridized with a two-point crossover operator (2 PCO FA) still outperforms the basic implementation, but does not provide better results than hybridization with a single-point operator.

In the last experiment, the authors decided to use the full potential of the implemented hybridization. Not only child 1 was used to substitute the worst, but a second child (child 2) was generated using the same crossover point, and it was used to substitute the second worst solution of the population (see Fig. 5). This approach prevents the potentially better solution of the crossover operation from being discarded.

TABLE III. BASIC FA VS. 2-CHILDREN HYBRID FA (2-C FA)

Performance Data			
FA Setup	Average	Worst	Best
Basic FA	3.90315E-08	1.04747E-07	1.61495E-08
2-C FA (d=6, p_t=0)	2.25077E-08	6.25977E-08	7.7884E-09
2-C FA (d=7, p_t=0)	2.20455E-08	3.87267E-08	5.889E-09
2-C FA (d=8, p_t=0)	2.28874E-08	5.10211E-08	1.04E-08

According to Table III, the results from the basic FA and the best previous setup (hybrid FA, (d=6, p_t=0)) show that by using both offspring solutions even better results can be attained. All runs (d=6…8) show a better on average performance than the current best solution. Also, the best solutions are better for each run than the best previous setup.

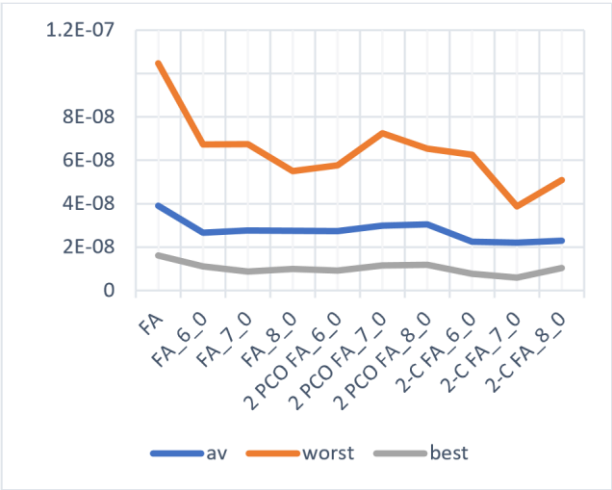

Figure 8. Performance of FA, hybrid FA, 2 PCO FA and 2-C FA.

Fig. 8 shows that while all hybrid approaches outperform the basic FA, the 2-children hybrid FA shows the best exploitation mechanism. It shows that a parameter setting of d=7, 8, p_t=0 using the 2-children hybrid FA directs the search in the most efficient way towards the optimal solution. This setup outperforms all others.

VI. CONCLUSIONS

In this paper, the suggested hybrid FA was classified as an integrative hybrid FA algorithm. Further, the effects of hybridizing the basic FA with the GA were investigated when enabling the hybridization on different stages applying various probabilities. A hybrid approach used in a 2-dimensional search space was not able to outperform the basic FA. In a more complex scenario, the hybrid FA was able to outperform the basic FA. It was shown that the hybrid FA performs best when the probability threshold p_t, is set to 0 and the hybridization is enabled at a stage above 0.8*N (d=6). It has also been discovered that it makes no difference whether a single-point or two-point crossover operator is applied to generate the mutant. Further, it shows that optimal results can be attained if both offspring of a crossover operation are used.

In future research, these findings can be used to solve real-world problems, for example portfolio optimization problems. It would be useful to test whether the performance of the hybrid FA can be improved even with more complex and realistic problems if the suggested parameters are applied and both offspring solutions are used.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Kevin Saner and Kyle Smith developed the hybrid FA, conducted the experiments, and wrote the first version of the paper. Thomas Hanne and Rolf Dornberger suggested and supervised the research and revised the paper. All authors had approved the final version.

REFERENCES

[1] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, "Metaheuristic algorithms in modeling and optimization," in *Metaheuristic Applications in Structures and Infrastructures*, Elsevier, 2013, pp. 1-24.

[2] X. S. Yang, S. Deb, T. Hanne, and X. He, "Attraction and diffusion in nature-inspired optimization algorithms," *Neural Computing and Applications*, vol. 31, no. 7, pp. 1987-1994, 2019.

[3] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd edition, Luniver Press, 2010.

[4] F. J. Rodriguez, C. Garcia-Martinez, and M. Lozano, "Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison, and synergy test," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 787-800, 2012.

[5] X. S. Yang, S. Cheng, and K. Huang, "Hybrid metaheuristic algorithms: Past, present, and future," *Recent Advances in Swarm Intelligence and Evolutionary Computation*, pp. 71-83, 2014.

[6] R. Dornberger, *Applied Computational Intelligence: 4. Evolutionary Computation and Genetic Algorithms*, University of Applied Sciences Northwestern Switzerland, Olten, 2020.

[7] X. S. Yang, "Test problems in optimization," arXiv preprint arXiv:1008.0549, 2010.

[8] K. Hussain, M. Najib, M. Salleh, and S. C. R. Naseem, "Common benchmark functions for metaheuristic evaluation: A review," *International Journal on Informatics Visualization*, vol. 1, no. 4-2, pp. 218-223, 2017.

[9] S. Surjanovic and D. Bingham. (2017). Optimization test problems. Simon Fraser University. [Online]. Available: https://www.sfu.ca/~ssurjano/optimization.html

[10] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimization," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78-84, 2010.

[11] M. Tuba and N. Bacanin, "Upgraded firefly algorithm for portfolio optimization problem," in *Proc. UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, 2014, pp. 113-118.
[12] S. Sefiane and M. Benbouziane, "Portfolio selection using genetic algorithm," *Journal of Applied Finance & Banking*, no. 2, pp. 143-154, 2012.
[13] I. Strumberger, E. Tuba, N. Bacanin, M. Beko, and M. Tuba, "Hybridized artificial bee colony algorithm for constrained portfolio optimization problem," in *Proc. IEEE Congress on Evolutionary Computation*, Lisbon, 2018, pp. 1-8.
[14] I. Strumberger, N. Bacanin, and M. Tuba, "Constrained portfolio optimization by hybridized bat algorithm," in *Proc. 7th International Conference on Intelligent Systems, Modelling and Simulation*, Bangkok, 2016, pp. 83-88.
[15] X. S. Yang. (2020). Firefly algorithm. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/29693-firefly-algorithm

Kevin Saner was born in Mümliswil, Switzerland in September 1991. After doing an apprenticeship as a machine mechanic, he obtained his bachelor degree in Systems Engineering from FHNW in Brugg, Switzerland in 2016. Currently, he is enrolled as a master student in the study programme Business Information System at FHNW in Olten, Switzerland. After graduating in 2016, he worked as a Software Engineer in logistics at Swisslog. Currently, he works as IT Analyst at Educa, a special agency for digitalizing the education system, in Bern. His research interests include machine learning and optimization algorithms.

Kyle Smith was born in Cape Town, South Africa in 1993 and grew up in Switzerland. After attending the vocational business school in St. Gallen - he received his Joint-Bachelor-Degree in Media Engineering in 2018 with a focus on application development from the FHGR and BFH in Bern. Presently he is attending the double degree master's program in Business Information Systems at the FHNW and Computer Science at the University of Camerino, Italy. He has worked as an analyst and project manager at the Swiss Confederation for the United Nation's Agenda 2030 sustainability program. Currently he is working as a Solutions Engineer at Siemens Smart Infrastructure in Zug. His research areas of interest are machine learning, human-computer-interaction, knowledge engineering, process mining, evolutionary algorithms and blockchain.
Kyle Smith is a former board member of the Erasmus Student Network (ESN) Bern and is supporting ESN Switzerland by improving conditions and programs for international student mobility.

Rolf Dornberger, born in Germany, studied Air- and Aerospace Engineering (diploma 1994) at universities in Stuttgart (Germany), Barcelona (Spain) and Grenoble (France). He holds a PhD (1998) in Air- and Aerospace Engineering from the University of Stuttgart, Germany, in the field of numerical methods for modelling and simulation, and an international University Teaching Certificate from the University of Basel, Switzerland (2017).
After his PhD, he worked in industry in different management positions as a Consultant, IT Officer and Senior Researcher in different international engineering, technology, and IT companies in the field of energy, software, IT, and airline business in Switzerland. He taught part-time at universities in Zurich and Stuttgart. Today, he is Professor and Head of the Institute for Information Systems at the School of Business of the University of Applied Sciences and Arts Northwestern Switzerland FHNW in Basel, Switzerland. Additionally, he is a guest lecturer at the Faculty of Business and Economics of the University of Basel, Switzerland. He is the (co-)author of more than a hundred scientific publications. His current research interests include artificial intelligence, particularly the nature-inspired methods of computational intelligence, modelling, simulation and optimization, robotics, human-machine interaction, and innovation management.
Prof. Dr. Dornberger is a member of Swiss Informatics and Rotary, has been vice president and board member of two Swiss associations, and has/had various commitments as a member of international and technical program committees, book author and editor (Springer) and journal editor, honorary chairman, session leader, organizer of special sessions, reviewer, invited keynote speaker.

Thomas Hanne received master's degrees in Economics and Computer Science, and a PhD in Economics. From 1999 to 2007 he worked at the Fraunhofer Institute for Industrial Mathematics (ITWM) as senior scientist. Since then, he is Professor for Information Systems at the University of Applied Sciences and Arts Northwestern Switzerland and Head of the Competence Center Systems Engineering since 2012.
Thomas Hanne is author of more than 160 journal articles, conference papers, and other publications and editor of several journals and special issues. His current research interests include computational intelligence, evolutionary algorithms, metaheuristics, optimization, simulation, multicriteria decision analysis, natural language processing, systems engineering, software development, logistics, and supply chain management.