Solving Agile Software Development Problems with Swarm Intelligence Algorithms

Lucija Brezočnik, Iztok Fister Jr. and Vili Podgorelec

Abstract This paper outlines a short overview of swarm intelligence algorithms that are used within the software engineering area. Swarm intelligence algorithms have been used in many software engineering tasks, e.g., grammatical inference or mutation testing. However, their presence in the agile software development field is still awakening. As there are some promising results of solving different problems of agile software development with swarm intelligence, this paper discusses such problems and the proposed solutions within the last decade. Based on the results we propose a systematic classification of swarm intelligence algorithms according to problems within agile software development, i.e., next release problem, risk, software design, software cost estimation, and software effort estimation. Afterwards, we present papers that fall in the scope of the proposed classification, and provide highlights of each paper for researchers, conducting research in this and associated fields. In this manner, we provide some conclusions for each of the classified problem groups, and, in the end, we review the guidelines for the future.

1 Introduction

Swarm intelligence or, simply, SI algorithms, are a sub-branch of Computational Intelligence. Loosely speaking, swarm intelligence algorithms are methods that are inspired mostly by nature. In other words, they concern the collective, emerging

Lucija Brezočnik
Faculty of Electrical Engineering and Computer Science, University of Maribor,
Koroška cesta 46, 2000 Maribor, Slovenia
e-mail: lucija.brezocnik@um.si

Iztok Fister Jr.
e-mail: iztok.fister1@um.si
· Vili Podgorelec
e-mail: vili.podgorelec@um.si

behavior of multiple, interacting agents who follow some simple rules [7]. These agents might be considered as unintelligent. Interestingly, when working together, the whole system of multiple agents may show some self-organization behavior (collective intelligence). The history of swarm intelligence goes back to 1989 when Beni [4] coined this term. Expansion of these algorithms began after the 1990s. On the one hand, researchers showed that they are very useful when solving continuous optimization problems, while, on the other, they also behave well in discrete optimization problems. Interestingly, there are also a lot of practical applications that are based on SI algorithms in the real world. During the past decades, many swarm intelligence algorithms were also applied in the domain of Software Engineering, along with evolutionary algorithms [12]. Mostly, researchers were concentrated on solving problems such as the development of mutation testing [13], grammatical inference [21], and test effort estimation [27]. In contrast, too little attention was devoted to the swarm intelligence algorithms in the domain of agile software development. According to our literature research, we saw that in recent years more applications had been proposed in the literature. For that reason, the primary missions of this paper are:

- to present a short overview of this vital research field,
- to review this research field and classify problems that are solved by swarm intelligence algorithms,
- to study why swarm intelligence algorithms are useful for solving problems within the agile software development research field, and
- to determine guidelines for the future of this research field.

The structure of this paper is as follows: Section 2 acquaints the reader with the fundamentals of swarm intelligence, while Section 3 presents agile software development methods. Section 4 discusses problems in agile software development that were tackled by swarm intelligence algorithms. Section 5 is devoted to the future of this field, while the paper is concluded with a summary of SI methods in the agile software development field.

2 Core fundamentals of swarm intelligence

Let us imagine bees when searching for nectar, or ants when building anthills, or even fireflies when mating during the summer nights. At first sight, we can say that they are pure individuals that would like to survive in their natural habitat. However, this observation is not correct. Although these individuals are considered as unintelligent, they cooperate in each aspect. These characteristics can be conceived as swarm intelligence. Swarm intelligence involves the collective, emerging behavior of multiple, interacting agents who follow some simple rules. While each agent may be considered as unintelligent, the whole system of multiple agents may show some self-organizing behavior and, thus, can behave as a kind of collective intelligence [7].

Nowadays, many algorithms have been developed by drawing inspiration from swarm intelligence systems. Roughly speaking, there are probably more than 100 SI algorithms, due to the popularity of SI research. However, some researchers have recently warned that some algorithms might have roots in existing algorithms [26]. Among the most well-established SI algorithms are: Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony Optimization (ABC), Firefly Algorithm (FA), Cuckoo Search (CS) and Bees Algorithms (BA).

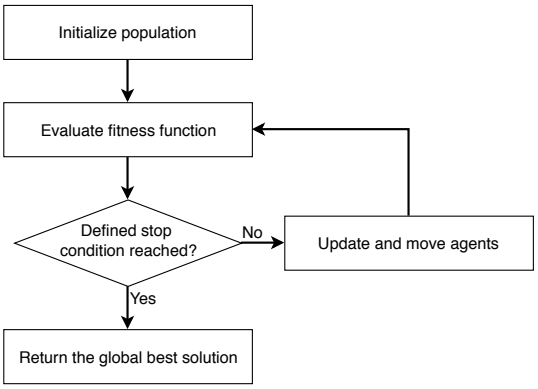

Fig. 1 Swarm intelligence framework [7].

Even though there is a bunch of SI algorithms, all of them follow the SI framework presented in Fig.1. All SI algorithms are population-based. Therefore, the first step in the algorithm is a random generation of the initial population and evaluation of this population. Later, in the main loop, the individuals in the search space are moved towards the best individuals, while the best-evaluated individuals are selected for the next generation.

3 Agile methods

Nowadays, we can hardly find non-agile software companies, i.e., companies that do not utilize agile practices in their product, the project development, or both. The main reason for that is because they want to accelerate product delivery, enhance the ability to manage changing priorities, and increase productivity [29]. Thus, we can see that the biggest problem of the traditional approaches is their incapability to respond to the constant flow of changes quickly even though this is the most important thing for the customers. Consequently, many agile methodologies were proposed and introduced in companies all around the world. The most frequently used are still Scrum and some custom Scrum hybrids (62%), followed by Scrumban (8%), and Kanban (5%) [29].

The question that arises here is why do we need to include swarm intelligence techniques into agile software development? The answer is quite simple. Firstly, agile software development is more than just code writing and testing. Many optimization problems occur already at the beginning of a project, i. e., planning. What are the functionalities that must be developed in the first iteration? How will we evaluate the effort of the tasks? How long will it take us to develop a specific task? These are just a few questions that project managers are dealing with daily. However, the advantage of those questions is that we can describe and present them as optimization problems. After the problem is described mathematically, we can tackle it with many SI algorithms to find the optimal solution for the given problem. Although optimal solutions are mostly hard to find because of the multiple conflicting objectives such as lack of data and benchmarks, the research on this field is awakening.

In Section 4, we present some solutions where researchers introduced SI algorithms to specific agile software design problem. Search methodology is defined in Subsection 4.1 and the obtained results in Subsection 4.2.

4 Methods and applications

4.1 Search Methodology

The methodology for searching the relevant literature for this paper is the following. Firstly, we defined the search term. We combined the most used SI algorithms (PSO, ACO, ABC, CS, BA, and CaSO) with the software development problem (NRP, R, SD, SCE, and SEE). For example, while searching for papers that applied the PSO algorithm to the next release problem, we used the following search term: ((”PSO” OR ”particle swarm optimization” OR ”particle swarm optimisation”) AND (”Next Release Problem”)).

The search was limited to the four major databases, i.e., Springer Link, IEEE Explore, Google Scholar, and ScienceDirect. Next step combined pre-screening of the results, where we eliminated redundant and inappropriate results. This approach resulted in 21 selected papers that are presented in detail in Subsection 4.2.

4.2 Results

Agile software development problems tackled with the SI algorithms can be divided into the following five groups:

Next Release Problem (NRP) is present in software development companies all the time. In this phase, features should be selected that must be developed in the next release. The selection process is very hard, since multiple constraints must

be taken into account, such as cost, time, dependent requirements, client satisfaction, reliability. The goal is to find the optimal solution for the given restrictions;

Risk (R) To satisfy requirements of quality software, risk factors must be well defined and prioritized to avoid any overpayment of costs or money. Thus, the authors try to apply techniques for risk factor prioritization;

Software Design (SD), where software designers try to find good designs of software in the early stages of the software development process. The authors tackle this problem with different interactive and non-interactive approaches;

Software Cost Estimation (SCE) is a process in which the required time and cost are predicted. When dealing with this kind of problem, the authors, in most cases, try to tune the parameters of the Constructive Cost Model (COCOMO);

Software Effort Estimation (SEE), as the name implies, is a process in which the amount of effort required to develop a product increment is estimated. Such estimations are, in literature, done in many cases with Case-Based Reasoning (CBR) and fuzzy logic for simulation of the uncertainty factors.

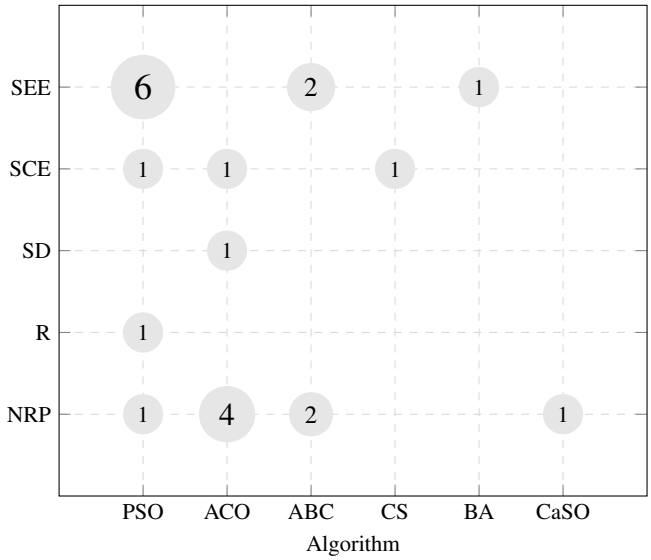

Fig. 2 Bubble plot for the number of papers regarding two variables: Algorithm (x-axis) and software development problem (y-axis).

Based on our literature research, only six different SI algorithms were applied to solve the mentioned problems, i.e., PSO, ACO, ABC, Cuckoo Search (CS), Bee Algorithm (BA), and Cat Swarm Optimization (CaSO). An overview of the findings, in which eleven journal articles, seven book chapters, and three papers from conference proceedings have been included, is presented in Tables 2– 4, while Fig. 2 presents the distribution of SI algorithms across different software development tasks, and

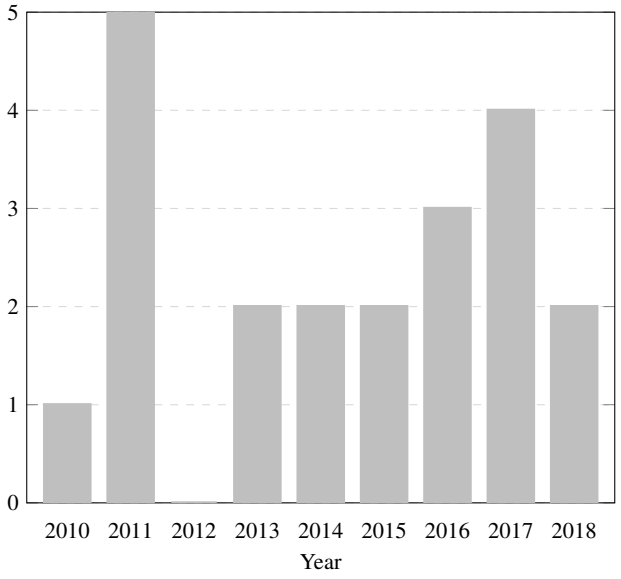

Fig. 3 Analyzed papers that appeared in a particular year.

Fig. 3 presents the dynamics of the occurrence of specific algorithms in the literature over time [1]. As we can see from Tables 2– 4, each paper is described with the base SI algorithm, the algorithms and/or methods used to compare the obtained results of the proposed SI algorithm with, and some remarks on the proposed SI algorithm for a given software development problem.

The papers addressing the NRP are listed in Table 2. The proposed SI algorithms, aimed at optimizing the selection of features to be developed in the next release, are focused primarily on finding the optimal solution while fulfilling all given constraints (time, interaction, cost, and budget thresholds, effort boundaries, constraints regarding requirements, Scrum task allocation). Within this category, ACO and ABC are used predominantly, while being primarily compared to genetic algorithms, simulated annealing, hill climbing, GRASP, and NSGA-II, as well as to manual optimization.

Table 1 lists the papers which address the SCE problems. When compared to NRP, the number of research papers within this category is much smaller. ACO, PSO, and CS algorithms have been used here to best estimate the required time and cost of software development projects. Interestingly, all the proposed SI algorithms within this category have been, besides to the COCOMO model and genetic programming, compared to other SI algorithms, which suggests that there is a lack of existing cost estimation methods available.

[1] Note that only the last nine years were considered in this study.

The papers listed in Table 3 address the SEE problems in software development. The PSO algorithm is used predominantly for problems within this category, followed by the ABC and BA algorithms. Similar to the SCE problems, also here, the fuzzy logic is applied in some cases to handle the uncertainty in effort estimation. Besides providing the absolute effort estimation, the SI algorithms are also used to reduce the difference between actual and predicted effort when using some other prediction methods or techniques. Within this category, the proposed SI algorithms are, in general, compared with the highest number of other existing effort estimation approaches, including the COCOMO model, analogy-based estimation, genetic algorithms, artificial neural networks, case-based reasoning, multiple regression, regression towards the mean, stepwise regression, as well as to a number of other SI algorithms, mostly ABC (and PSO, if the proposed algorithm is not based on it).

Finally, Table 4 lists the papers which address some other software development problems (SD, R). The research on these topics is scarce, as we have been able to find only two such papers. Like other software development problems, however, also here the two most commonly used SI algorithms have been used – PSO and ACO. They have both been compared to methods from the decision analysis area, which combine objective and subjective measures to find a solution that best utilizes the given goals.

To summarize the most important findings, we see that not many SI algorithms were applied to solve software development problems. Out of those algorithms which were, PSO still predominates (39%), followed closely by ACO (33%) (See Fig. 2). However, regardless of the used algorithm, it seems that, for uncertainties in software development problems (especially in SEE and SCE) authors simulate mostly with the use of fuzzy logic.

Table 1 Overview of the SI algorithms used for the SCE problems

Ref	Base algorithm	Compared to	Remarks
[28]	ACO	PSO, GP, RMSE	Proposed algorithm combined with TSP for SCE. Results were evaluated with three datasets in terms of Root Mean Square Error (RMSE).
[16]	CS	COCOMO, KNN, CUCKOO-KNN	Proposed fuzzy inference system combined with Cuckoo optimization algorithm. Results were evaluated with tera-PROMISE datasets in terms of improved accuracy and cost estimation.
[24]	PSO	COCOMO	Proposed model for COCOMO parameters' tuning using multi-objective PSO with objectives (mean absolute relative error and prediction). Results were evaluated with the COCOMO dataset.

* KNN–k-Nearest Neighbors; TSP–Traveling Salesman Problem

Table 2 Overview of the SI algorithms used for the NRP problems

Ref	Base algorithm	Compared to	Remarks
[23]	ABC	MOTLBO	Proposed algorithm with objectives (minimum cost, maximum client satisfaction, minimum time consumption and maximum reliability) and constraints (time threshold, interaction and cost threshold). Results were evaluated in terms of hyper-volume indicator, spread indicator and number of non-dominated solutions.
[8]	ABC	ACO, NSGA-II, GRASP	Proposed algorithm with objectives (cost and satisfaction) and constraints (types of interaction). Results were evaluated with two real life datasets with the interms of hyper-volume indicator, spread indicator and number of non-dominated solutions.
[11]	ACO	NIACS	Proposed single-objective formulation for the interactive version of the NRP with the budget constraints and incorporated user preferences. Results were evaluated with three real life datasets in terms of budget constraints and user preferences.
[10]	ACO	GRASP, NSGA-II	Proposed multi-objective ACS for requirements' selection. Results were evaluated with two real life datasets in terms of hyper-volume indicator, spread and spacing indicators, and number of non-dominated solutions. Highlighted were problems with crossover and mutation operations in NSGA-II in NRP.
[9]	ACO	GA, SA	Proposed method for the NRP problem with dependent requirements. Results were evaluated with the 72 synthetic datasets in terms of quality and execution time.
[14]	ACO	GA, SA, FHC, ACO	Proposed hybrid ACO method with incorporated local search to improve solution quality. Results were evaluated with five synthetic datasets in terms of solution quality and execution time.
[15]	CaSO	synthetic dataset	Proposed multi-objective collaborative scheduling model for NRP. Results were evaluated with dataset in terms of product development time and costs.
[5]	PSO	manual allocation	Proposed method for Scrum task allocation problem with constrains. Results were evaluated with the real life internal project.

* FHC–First Found Hill Climbing; SA–Simulated Annealing; CaSO–Cat Swarm Optimization

Table 3 Overview of the SI algorithms used for the SEE problems

Ref	Base algorithm	Compared to	Remarks
[19]	ABC	ABC, COCOMO II	Proposed algorithm with the teaching-learning mechanism applied to the ABC algorithm. Results were evaluated with a NASA software project dataset in terms of SEE. Highlighted faster convergence than ABC.
[18]	ABC/ PSO	ABC, PSO, regression	Proposed method based on velocity and story point factors, where parameters are optimized using PSO. Results were evaluated on dataset in terms of accuracy of predicted results.
[3]	BA	CBR, GA, RTM, other papers	Proposed method to adjust the retrieved project efforts and find the optimal number of analogies by using BA. Results were evaluated on six datasets in terms of different performance measures. Search capability of the BA applied to overcome the local tuning problem of effort adjustment.
[17]	PSO	CART, SWR, MLR, ANN, ABE	Proposed Analogy-Based Estimation (ABE) algorithm combined with PSO. Results were evaluated with the three real life datasets in terms of accuracy of the SEE.
[30]	PSO	CBR methods	Proposed optimized weights of CBR methods with PSO. Results were evaluated with the two datasets in terms of three quality metrics, i.e., mean magnitude of relative error, median magnitude of relative error and Pred(0.25).
[2]	PSO	UCP, TPA	PSO applied to UCP and TPA to reduce the difference between actual and predicted effort. Results were evaluated with the cases from two papers and compared regarding UCP or TPA.
[22]	PSO	SEE models	Proposed algorithm that applies fuzzy logic to obtain uncertainty in EE and PSO for parameters' tuning. Results were evaluated with ten NASA software projects on the basis of the VAF, MARE, and VARE.
[20]	PSO	COCOMO	Proposed algorithm for COCOMO parameters' tuning using multi-objective PSO. Results were evaluated with Magnitude of relative error and prediction level.

* ANN–Artificial Neural Network; CBR–Case-Based Reasoning; MLR–Multiple regression; RTM–Regression Towards the Mean; SWR–Stepwise Regression; UCP–Use Case Points; TPA–Test Point Analysis; VAF–Variance accounted For; MARE–Mean Absolute Relative Error; VARE–Variance Absolute Relative Error

Table 4 Overview of the SI algorithms used for some other (SD, R) software development problems

Ref	Base alg.	Compared to	Remarks
[25]	ACO	IEA	SD problem. Proposed multi-objective ACO search steered jointly by an adaptive model that combines subjective and objective measures. Results were evaluated by the experts.
[1]	PSO	AHP	R problem. Proposed method for optimization of the project duration by using a current optimal risk factor with PSO. Results were evaluated with ten agile software development projects.

* AHP–Analytic Hierarchy Process

5 Future paths

Although the research in the software development area was more focused on the GA algorithms in the past, some problems arise regarding the GA fundamental phases. According to the authors in [10], the most obvious problems using GA are crossover and mutation operations, especially in the NRP when considering restrictions. Therefore, researchers try to find some other ways to solve software development problems. Recently, research using SI algorithms for software development problems has increased. Nevertheless, there are still some challenges that must be addressed.

One of the most significant problems is test data. The optimal way is to use data that were obtained from a real-life scenario, but we often do not have such access. Therefore, online repositories should be prepared with multiple projects. For each project, parameters should be defined, such as the number of iterations, requirements, dependencies between requirements, planned vs. actual software development process (can also be in the form of a Burndown chart). Furthermore, projects could be classified by difficulty, e.g., projects with multiple dependencies are harder to solve than those with fewer. If real-life data could not be obtained, a dataset generator for the systematic generation of instances should be provided, as was also highlighted by the authors in [8].

Benchmarks for each project could be defined if we refer to repositories. With such benchmarks, we could facilitate the work of researchers who propose some novel algorithm and want to check given solutions briefly.

As far as the algorithms themselves are concerned, other SI algorithms should be applied to the already mentioned problems. After that, a study can be conducted with an emphasis on which of the SI algorithms performed the best for the specific software development problem. Moreover, it would also be sensible to check various hybridization of the SI algorithms, as was pointed out by Kuhat and Thi My Hanh [19]. With hybridization, we can take advantage of the powerful features of more than one individual algorithm and find potentially better solutions.

A subfield worth exploring is also class distribution skews and underrepresented data in software defect prediction [6].

6 Conclusion

Agile software development is present in many software development companies worldwide. Software companies make use of various agile methods to improve the productivity of teams and write better code with fewer bugs. Recently, some researchers even improved these methods by combining them with artificial intelligence methods.

In this paper, we made a short overview of swarm intelligence algorithms that are applied within the agile software development field. The systematic outline shows that swarm intelligence methods are beneficial for solving agile software development tasks. In line with this, we can expect more solutions that are based on swarm intelligence algorithms in the future.

Acknowledgements The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

References

1. R. Agrawal, D. Singh, and A. Sharma. Prioritizing and optimizing risk factors in agile software development. In *2016 Ninth International Conference on Contemporary Computing (IC3)*, pages 1–7, 2016.
2. S. Aloka, P. Singh, G. Rakshit, and P. R. Srivastava. *Test Effort Estimation-Particle Swarm Optimization Based Approach*, pages 463–474. Springer Berlin Heidelberg, 2011.
3. M. Azzeh. *Adjusted Case-Based Software Effort Estimation Using Bees Optimization Algorithm*, pages 315–324. Springer Berlin Heidelberg, 2011.
4. G. Beni and J. Wang. *Swarm Intelligence in Cellular Robotic Systems*, pages 703–712. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.
5. L. Brezočnik, I. Fister, and V. Podgorelec. Scrum task allocation based on particle swarm optimization. In P. Korošec, N. Melab, and E.-G. Talbi, editors, *Bioinspired Optimization Methods and Their Applications*, pages 38–49, Cham, 2018. Springer International Publishing.
6. L. Brezočnik and V. Podgorelec. Applying weighted particle swarm optimization to imbalanced data in software defect prediction. In I. Karabegović, editor, *New Technologies, Development and Application*, pages 289–296, Cham, 2019. Springer International Publishing.
7. L. Brezočnik, I. Fister, and V. Podgorelec. Swarm intelligence algorithms for feature selection: A review. *Applied Sciences*, 8(9), 2018.
8. J. M. Chaves-González, M. A. Pérez-Toledano, and A. Navasa. Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. *Knowledge-Based Systems*, 83:105–115, 2015.
9. J. T. de Souza, C. L. B. Maia, T. d. N. Ferreira, R. A. F. d. Carmo, and M. M. A. Brasil. *An Ant Colony Optimization Approach to the Software Release Planning with Dependent Requirements*, pages 142–157. Springer Berlin Heidelberg, 2011.
10. J. del Sagrado, I. M. del Águila, and F. J. Orellana. Multi-objective ant colony optimization for requirements selection. *Empirical Software Engineering*, 20(3):577–610, 2015.

11. T. do Nascimento Ferreira, A. A. Arajo, A. D. B. Neto, and J. T. de Souza. Incorporating user preferences in ant colony optimization for the next release problem. *Applied Soft Computing*, 49:1283–1296, 2016.
12. M. Harman. The current state and future of search based software engineering. In *2007 Future of Software Engineering*, pages 342–357. IEEE Computer Society, 2007.
13. Y. Jia and M. Harman. An analysis and survey of the development of mutation testing. *IEEE transactions on software engineering*, 37(5):649–678, 2011.
14. H. Jiang, J. Zhang, J. Xuan, Z. Ren, and Y. Hu. A Hybrid ACO algorithm for the Next Release Problem. In *The 2nd International Conference on Software Engineering and Data Mining*, pages 166–171. IEEE, 2010.
15. J.-j. Jiang, X. Yang, and M. Yin. Cooperative control model of geographically distributed multi-teamagile development based on mo-cso. In *Proceedings of the 2Nd International Conference on E-Education, E-Business and E-Technology*, ICEBT 2018, pages 121–125, New York, NY, USA, 2018. ACM.
16. A. Kaushik, S. Verma, H. J. Singh, and G. Chhabra. Software cost optimization integrating fuzzy system and COA-Cuckoo optimization algorithm. *International Journal of System Assurance Engineering and Management*, 8(2):1461–1471, 2017.
17. V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi. A PSO-based model to increase the accuracy of software development effort estimation. *Software Quality Journal*, 21(3):501–526, 2013.
18. T. Khuat and M. Le. A Novel Hybrid ABC-PSO Algorithm for Effort Estimation of Software Projects Using Agile Methodologies. *Journal of Intelligent Systems*, pages 1–18, 2017.
19. T. Khuat and L. Thi My Hanh. Applying teaching-learning to artificial bee colony for parameter optimization of software effort estimation model. *Journal of Engineering Science and Technology*, 12(5):1178–1190, 2017.
20. I. Manga and N. Blamah. A particle Swarm Optimization-based Framework for Agile Software Effort Estimation. *The International Journal Of Engineering And Science (IJES)*, 3:30–36, 2014.
21. M. Mernik, D. Hrnčič, B. R. Bryant, A. P. Sprague, J. Gray, Q. Liu, and F. Javed. Grammar inference algorithms and applications in software engineering. In *International Symposium on Information, Communication and Automation Technologies, 2009. ICAT 2009. XXII*, pages 1–7. IEEE, 2009.
22. P. V. G. D. Prasad Reddy and C. V. M. K. Hari. *Fuzzy Based PSO for Software Effort Estimation*, pages 227–232. Springer Berlin Heidelberg, 2011.
23. N. Ranjith and A. Marimuthu. A Multi Objective Teacher-Learning-Artificial Bee Colony (MOTLABC) Optimization for Software Requirements Selection. *Indian Journal of Science and Technology*, 6, 2016.
24. G. S. Rao, C. V. P. Krishna, and K. R. Rao. *Multi Objective Particle Swarm Optimization for Software Cost Estimation*, pages 125–132. Springer International Publishing, 2014.
25. C. L. Simons, J. Smith, and P. White. Interactive ant colony optimization (iACO) for early lifecycle software design. *Swarm Intelligence*, 8(2):139–157, 2014.
26. K. Sörensen. Metaheuristics–the metaphor exposed. *International Transactions in Operational Research*, 22(1):1–16, 2013.
27. P. R. Srivastava, A. Varshney, P. Nama, and X.-S. Yang. Software test effort estimation: a model based on cuckoo search. *International Journal of Bio-Inspired Computation*, 4(5):278–285, 2012.
28. V. Venkataiah, R. Mohanty, J. S. Pahariya, and M. Nagaratna. *Application of Ant Colony Optimization Techniques to Predict Software Cost Estimation*, pages 315–325. Springer Singapore, 2017.
29. VersionOne. *VersionOne 12th Annual State of Agile Report*. 2018.
30. D. Wu, J. Li, and Y. Liang. Linear combination of multiple case-based reasoning with optimized weight for software effort estimation. *The Journal of Supercomputing*, 64(3):898–918, 2013.