Artificial Bee Colony with Crossover Operations for Discrete Problems

Abdul Hadi Alaidi
Programming Department
Computer Science and Information Technology College
Wasit University
Wasit, Iraq
alaidi@uowasit.edu.iq

Soong Der Chen
College of Computing and Informatics
Universiti Tenaga Nasional
Selangor, Malaysia
chensoong@uniten.edu.my

Yeng Weng Leong
College of Engineering
Universiti Tenaga Nasional
Selangor, Malaysia
ywleong@uniten.edu.my

Received: 8 August 2022 | Revised: 27 August 2022 | Accepted: 29 August 2022

Abstract-The Artificial Bee Colony (ABC) is an algorithm designed to solve continuous problems. ABC has been proven to be more effective than other biological-inspired algorithms. However, it is needed to modify its functionality in order to solve a discrete problem. In this work, a natural modification to the original ABC is made to make it able to solve discrete problems. Six neighborhood operators are proposed to simulate the original behavior of ABC. Moreover, several Traveling Salesman Problem Library (TSPLIB) problems were used to examine the proposed method. The results of the proposed method are promising.

Keywords-Artificial Bee Colony (ABC); discrete problem; TSP

I. INTRODUCTION

A heuristic is a method for solving a problem that does not guarantee convergence to a global optimum but can offer a good enough solution. There are two types of heuristics for addressing a combinatorial optimization problem: perturbative and constructive heuristics [1]. A constructive heuristic creates a comprehensive solution from the bottom-up. A perturbative heuristic, on the other hand, changes an existing solution in order to find better solutions in the neighborhood of the present solution (i.e. operates neighborhood search operations). Problem-specific heuristics are used to solve particular problems. The nearest neighborhood heuristic and the multiple fragment heuristic are examples of constructive heuristics for the Traveling Salesman Problem (TSP), whereas perturbative heuristics include swap mutation, insert mutation, reverse mutation, order crossover, and partially mapped crossover.

A problem of the ABC algorithm is that it is originally designed for continuous problems so modification should be applied to make it work with other problem types. `ABC integrating grenade explosion and Cauchy operator to avoid random search and solving the Dynamic Economic Emission Dispatch (DEED) problem was established in [2]. Authors in [3, 4] used ABC for routing and performance enhancement of wireless sensor networks. Authors in [5] modified ABC to solve constrained optimization problems. Instead of the greedy selection of the original ABC, the modified ABC used Deb's rules for constrained problems. MABC is a Modified ABC for solving the stage shop scheduling problem [6]. The MABC uses tabu search in the employee phase instead of the greedy selection. Moreover, the onlooker bee phase was also modified and the idea of Particle Swarm Optimization (PSO) was used instead of random search. Authors in [7] analyzed the performance of a discrete ABC algorithm with a neighborhood operator. The original ABC uses two solutions for neighborhood search while the discrete ABC uses one solution for the neighborhood search.

The aim of this paper is to simulate the original neighborhood search behavior of ABC. The main contribution of this work is the use of a crossover operation as a neighborhood search and the introduction of six neighborhood operators which simulate the original ABC neighborhood searching behavior.

II. THE TRAVELING SALESMAN PROBLEM

The TSP is one of the most commonly investigated optimization problems. The statement of TSP is simple however, it remains a challenging problem. The TSP can be defined as finding the shortest path, for a given set of cities, starting from a base city, visiting all other cities once, and returning to the base city. The description of the TSP is an indirect weighted graph $G = (S, L)$, where S symbolizes the set of cities, and L symbolizes the set of links that connect the cities. Each link which connects two cities is assigned a length

Corresponding author: Abdul Hadi Alaidi

value d_{ij}, where $i, j \in L$. The problem is called symmetric TSP when the distance between two cities is the same regardless of direction, i.e. $d_{ij} = d_{ji}$. On the other hand, it is called asymmetric TSP when there is at least one link with a different distance with respect to direction, that is, the distances are direction-dependent and $d_{ij} \neq d_{ji}$. TSP is NP-hard problem. Equation (1) can be used to find the total possible solutions [8]:

$$Total\ solutions = \frac{(n-1)!}{2} \quad (1)$$

where n is the number of nodes or cities.

For example, if we have 60 cities and each city is connected to all other cities, the total possible solutions will be $\frac{(60-1)!}{2} = 6.9341559e+79$. In the case of evaluating 1 billion possible solutions per second, it would take about 2.2×10^{63} years to check all possible solutions [9]. From the above example, it is obvious how large the search space is. Although good techniques are used to reduce the search space (e.g. nearest neighbor [11]), the TSP is still classified as an NP-hard problem.

The TSPLIB benchmark [10] provides a large collection of TSP cases and is represented as a standard coordinate data system. Figures 1, 2 show an example of a TSP before finding the solution and the optimal solution respectively.

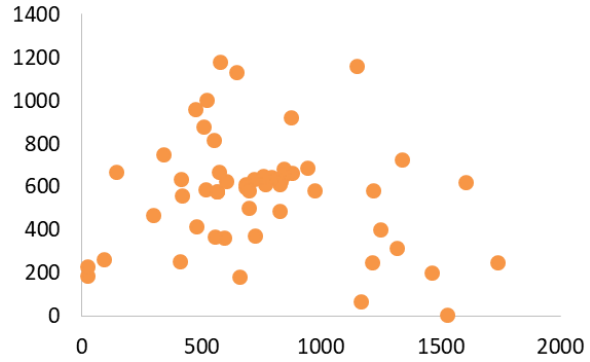

Fig. 1. A TSP problem.

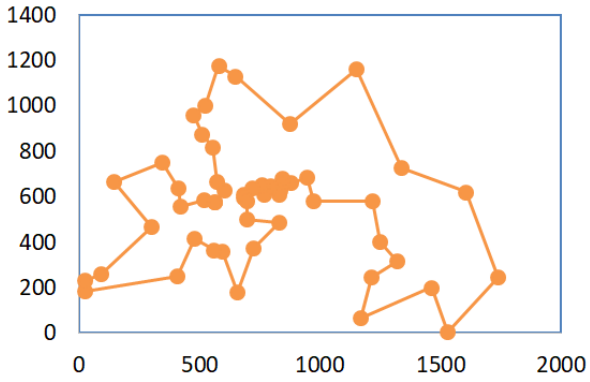

Fig. 2. TSP solution.

III. THE ARTIFICIAL BEE COLONY ALGORITHM

The ABC algorithm is a well-studied algorithm developed to solve continuous function optimization problems [11]. It simulates the foraging behavior a swarm of bees perform to find food. ABC is successfully used in many fields like energy management of for mobile devices [12] and for routing of mobile agents in Internet of Things (IoT) applications [13]. In ABC, the bees are divided into 3 groups: employed, onlookers, and scouts. Employed bees search for food sources and share information with an onlooker bee. The onlooker bee tries to search for a better food source in the neighborhood by using the knowledge passed from the employed bees. If the food source is abandoned, the employed bees responsible for that food source become scout bees and randomly search for the new food sources. In ABC, employed and onlooker bees accomplish the exploitation, while scout bees control the exploration process. In ABC, the employed bees use (2) to search neighbors by moving from an old position x_{ij} to the new position v_{ij} if and only if the new position's fitness is better than the fitness of the old position:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (2)$$

Where x_i is the old position and v_i is the new position of i^{th} employed. ϕ_{ij} is a random real number in the range of [-1,1], j is an arbitrary integer number between 1 and the dimension of the problem, while k is a random number between 1 and the number of employed bees. On the other hand, the onlooker bee selects a food source and evaluates all employed bees depending on the probability calculated by:

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \quad (3)$$

where *fit* is the fitness values of the solution (food source) of the employed bee, which is calculated by:

$$fit_i = \begin{cases} \frac{1}{1+f(x_i)}, & f(x_i) \geq 0 \\ 1 + |f(x_i)|, & f(x_i) < 0 \end{cases} \quad (4)$$

where $f(x_i)$ stands for the value of the objective function to be optimized. Like the employed bee, the onlooker bee uses (3) to update the position and explore the selected food source.

Algorithm 1 The standard artificial bee colony (ABC).

1. Initialization Phase
2. REPEAT
3. Employed Bees Phase
4. Onlooker Bees Phase
5. Scout Bees Phase
6. Memorize the best solution achieved so far
7. UNTIL(Cycle=Maximum Cycle Number or a Maximum CPU time)

Fig. 3. Standard ABC algorithm.

When a food source cannot be further exploited within a pre-defined trial limit, it will be abandoned, and the corresponding employed bee becomes a scout bee and randomly searches for a new food source. Such a combination of the exploration and exploitation search in ABC is important. It allows the algorithm to search the solution space for high-

quality solutions and prevents them from getting stuck in local optima. In ABC, the employed bees maintain the current solution while the onlooker bees allow the best solution to be exploited. Furthermore, scout bees try to remove stagnating solutions and explore new solutions. Figure 3 shows the standard ABC algorithm.

IV. ARTIFICAL BEE COLONY FOR TSP

TSP is a discrete problem and ABC is designed to solve continuous problems. The neighborhood operator which is a replacement for (2) is used in [7, 14] to make ABC able to solve TSPs. However, in the original ABC, the process of updating the solution is done by choosing another solution (randomly from the employee bee and using a roulette wheel in onlooker bee) and trying to enhance a solution. In this work, six neighborhood operators are proposed to simulate the original behavior of ABC.

A. Random Swap Crossover (RSC)

Random swap involves swapping the position of two randomly selected cities, starting from the first solution to the second solution, and rearranging the final solution to fix the position of the cities inserted. Consider, for example, the following two solutions of the 7-city TSP(1,2,3,4,5,6,7) and (1,5,7,6,2,3,4). First, select two random points as i and j. In our example $i = 3$ and $j = 5$. Then copy the i-th item from the second solution into the first solution (7→3). Then, the repair algorithm runs to map the i-th item in the first solution to the position of the old i-th item (3→7) and repeats the same operation for the j-th item as shown in Figure 4.

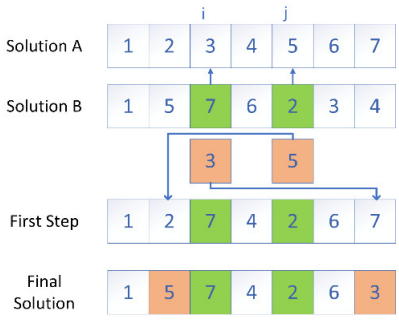

Fig. 4. Random Swap Crossover

B. Random Swap Subsequence Crossover (RSSC)

RSSC operation is the same as that of RSC, with one difference, which is the subsequence to swap in a solution from solution to solution. For example, if we have two solutions like in Figure 4 where $i = 2$ is the start position of sequence and $k = 3$, where k is the sequence length. First, copy the i-th item from the second solution into the first solution (5→2). Then, the repair algorithm maps the i-th item in the first solution to the position of the old i-th item (2→5) and repeats the same operation k times.

C. Random Insertion Crossover (RIC)

In RIC, a city from the first solution is randomly picked, removed from the second solution, reinserted into a random position in the second solution, and the repair algorithm is executed as explained in RSC. Figure 6 shows the RIC operation.

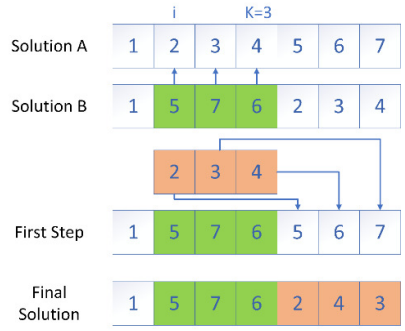

Fig. 5. Random Swap Subsequence Crossover.

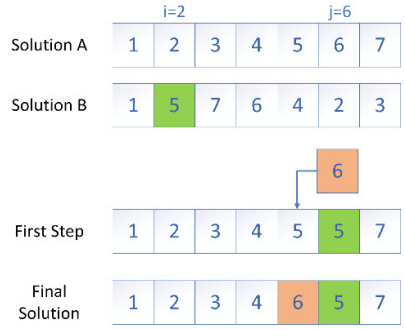

Fig. 6. Random Insertion Crossover.

D. Random Insertion of Subsequence Crossover (RISC)

In RISC, a subsequence from a solution is randomly picked and then inserted into a second solution. The repair algorithm is again executed as in RSC. Figure 7 shows the RISC operation.

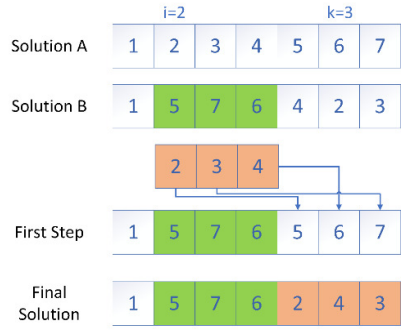

Fig. 7. Random Insertion of Subsequence Crossover.

E. Random Reversing Crossover (RRC)

RRC is the same as RSC, with one difference, which is the inversion of the inserted cities when copying from the first to the second solution as shown in Figure 8.

F. Random Swap Subsequence Reverse Crossover (RSSRC)

It is the same as RISC with the difference that the inserted sequence is inverted before insertion as shown in Figure 9.

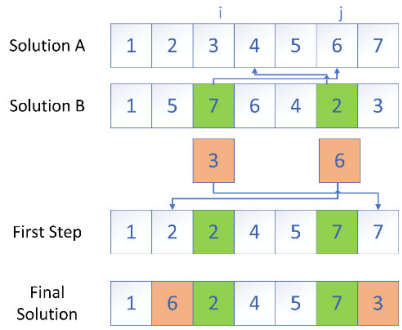

Fig. 8. Random Reversing Crossover.

Fig. 9. Random Swap Subsequence Reverse Crossover.

V. RESULTS AND DISCUSSION

In this work, the proposed method was implemented using C#. The algorithm was tested in 12 TSP problems (pr76, kroA100, kroB100, kroC100, kroD100, kroE100, rd100, gr120, pr124, bier127, ch130, pr136) taken from the TSPLIB benchmark library [12]. The code was run on a 2.80 GHz Intel Core i7 processor with 16 GB RAM. The same parameter values were used in all experiments to make sure of a fair judgment between all different methods. Each method ran the same TSP benchmark 30 times. The best and the average solutions were collected for every 30 runs.

The experimental results obtained by the discrete ABC algorithm with crossover and ABC with neighborhood operators explained in [7] are shown in Table I (RSC), Table II (RSSC), Table III (RIC), Table IV (RISC), Table V (RRC), and Table VI (RSSRC).

TABLE I. RANDOM SWAP CROSSOVER

Problem	RSWC		RS	
	error	avg error	error	avg error
pr76	46.99748	54.72502	73.58703	91.39779
kroA100	80.78658	96.99104	139.8036	163.6677
kroB100	79.63055	88.92176	135.0029	152.0174
kroC100	74.64938	98.85922	131.963	163.369
kroD100	85.98666	94.72684	136.2168	155.6664
kroE100	71.01233	92.52387	132.8938	153.7324
rd100	69.38053	85.36747	116.7762	145.2078
gr120	78.45001	94.11505	138.0726	154.8065
pr124	136.2697	168.2421	241.074	265.6827
bier127	47.11283	69.60656	105.9739	118.8277
ch130	86.30115	104.0835	161.9804	183.4746
pr136	87.92729	112.748	181.9473	197.9853

TABLE II. RANDOM SWAP SUBSEQUENCE CROSSOVER

Problem	RSSWC		RSS	
	error	avg error	error	avg error
pr76	59.73151	71.36817	25.39687	47.79393
kroA100	86.09153	108.4756	84.81346	110.7512
kroB100	77.17357	101.3199	78.25753	100.5301
kroC100	98.665	111.3994	90.92968	118.8134
kroD100	85.83639	103.1306	82.44576	105.7127
kroE100	90.4749	102.0346	80.2021	106.9497
rd100	83.67889	97.02318	74.71555	98.79351
gr120	85.07635	95.13253	88.61999	113.2892
pr124	131.1164	149.0558	138.3534	185.6913
bier127	69.38587	78.58206	52.21082	70.05222
ch130	107.234	120.3644	97.10311	122.9116
pr136	102.9296	115.7102	114.7532	142.3517

TABLE III. RANDOM INSERTION CROSSOVER

Problem	RIWC		RI	
	error	avg error	error	avg error
pr76	27.94312	39.36553	10.94685	20.9037
kroA100	38.01334	61.09639	35.77671	52.17602
kroB100	37.36507	55.50231	33.21892	51.09405
kroC100	52.78327	65.16459	30.12193	57.67137
kroD100	47.56269	60.26111	35.18832	53.36621
kroE100	44.0638	59.97311	31.6431	52.00773
rd100	31.66877	55.39022	31.32743	58.60598
gr120	48.2858	60.77883	53.31317	72.07241
pr124	68.42453	89.73748	63.58123	101.3253
bier127	41.45855	49.50531	47.90162	69.3766
ch130	57.54501	70.97709	74.04255	116.2635
pr136	54.72967	68.41197	77.31782	104.5502

TABLE IV. RANDOM INSERTION OF SUBSEQUENCE CROSSOVER

Problem	RISWC		RIS	
	error	avg error	error	avg error
pr76	49.2266	58.47	50.86031	66.37531
kroA100	76.14416	86.6875	104.2618	130.5938
kroB100	69.49099	81.44588	90.89472	113.5918
kroC100	79.03031	90.92808	100.8964	130.3549
kroD100	67.30065	81.70236	93.09665	119.8395
kroE100	64.37375	81.873	99.69639	124.8651
rd100	70.0885	81.38938	94.96839	115.2503
gr120	66.30654	77.97273	90.01729	121.2134
pr124	101.5297	117.2478	137.2438	200.9112
bier127	54.447	69.39636	67.83112	81.2194
ch130	79.1653	93.60993	111.9476	146.4146
pr136	74.26735	87.27759	135.192	156.8362

TABLE V. RANDOM REVERSING CROSSOVER

Problem	RRWC		RR	
	error	avg error	error	avg error
pr76	21.22307	26.50754	8.847839	13.58325
kroA100	18.54547	29.62665	8.4486	14.70384
kroB100	19.15563	27.48035	4.879309	12.21972
kroC100	6.691075	17.47819	1.227822	4.689023
kroD100	13.33333	18.1679	1.185185	4.019753
kroE100	15.1942	26.48397	15.15869	20.62213
rd100	25.02455	34.71413	28.01964	40.71358
gr120	19.04861	30.61965	4.089297	13.66924
pr124	16.38715	23.30662	5.198294	10.8402
bier127	32.74267	44.25733	17.94681	30.11028
ch130	26.09536	34.55156	22.58505	34.33286
pr136	21.25158	28.10493	8.609355	15.42815

The results clearly show that the proposed method is better for all considered problems and RSC, RISC, RRSSRC

methods. In RSSC, it got a better result in 9 cases. On the other hand, RIC solves half of the problems with better results than the other methods. However, the RRC operator gives poor performance compared to the random-reverse neighborhood operator.

TABLE VI. RANDOM SWAP SUBSEQUENCE REVERSE CROSSOVER

Problem	RSSRWC		RSSR	
	error	avg error	error	avg error
pr76	58.71911	70.09199	102.3012	114.1171
kroA100	93.22902	113.2607	182.3983	192.9061
kroB100	84.36837	103.8512	168.6645	178.0544
kroC100	85.88848	111.2542	169.8588	194.1552
kroD100	81.38443	105.0621	168.5404	185.8098
kroE100	90.64709	104.1044	169.0955	182.4486
rd100	84.93047	103.8972	168.8748	183.0523
gr120	85.86863	100.5695	161.8986	180.8902
pr124	131.9532	156.3919	283.1865	311.2413
bier127	67.18267	82.61759	125.0528	135.4019
ch130	107.2831	123.7621	212.455	223.1844
pr136	103.5847	120.2145	208.8197	229.9761

VI. CONCLUSIONS

In this work, the Artificial Bee Colony algorithm is modified with new neighborhood operators to solve discrete problems. The proposed method was tested on 12 TSP problems from the TSPLIB. The results show that Random Swap Crossover, Random Insertion of Subsequence Crossover, and Random Swap Subsequence Reverse Crossover operations give a good performance. The proposed method is similar to the original Artificial Bee Colony algorithm and gives promising results.

Our future work will focus on introducing new neighborhood operators and optimizing the implementation of the proposed method. Furthermore, due to the encouraging result, the proposed method can be easily integrated into other optimization algorithms.

ACKNOWLEDGMENTS

The authors acknowledge the financial support from the Universiti Tenaga Nasional under UNITEN-Iraq scholarships UNITEN BOLD research grant (J510050002/2021150).

REFERENCES

[1] Q. Gu, Q. Wang, X. Li, and X. Li, "A surrogate-assisted multi-objective particle swarm optimization of expensive constrained combinatorial optimization problems," *Knowledge-Based Systems*, vol. 223, Jul. 2021, Art. No. 107049, https://doi.org/10.1016/j.knosys.2021.107049.

[2] I. Marouani, A. Boudjemline, T. Guesmi, and H. H. Abdallah, "A Modified Artificial Bee Colony for the Non-Smooth Dynamic Economic/Environmental Dispatch," *Engineering, Technology & Applied Science Research*, vol. 8, no. 5, pp. 3321–3328, Oct. 2018, https://doi.org/10.48084/etasr.2098.

[3] S. D. Chavan and A. V. Kulkarni, "Event Based Clustering Localized Energy Efficient Ant Colony Optimization (EBC_LEE-ACO) for Performance Enhancement of Wireless Sensor Network," *Engineering, Technology & Applied Science Research*, vol. 8, no. 4, pp. 3177–3183, Aug. 2018, https://doi.org/10.48084/etasr.2121.

[4] H. Ehteshami, S. Javadi, and S. M. Shariatmadar, "Improving the Power Quality in Tehran Metro Line-Two Using the Ant Colony Algorithm," *Engineering, Technology & Applied Science Research*, vol. 7, no. 6, pp. 2256–2259, Dec. 2017, https://doi.org/10.48084/etasr.1551.

[5] D. Karaboga and B. Akay, "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems," *Applied Soft Computing*, vol. 11, no. 3, pp. 3021–3031, Apr. 2011, https://doi.org/10.1016/j.asoc.2010.12.001.

[6] M. M. Nasiri, "A modified ABC algorithm for the stage shop scheduling problem," *Applied Soft Computing*, vol. 28, pp. 81–89, Mar. 2015, https://doi.org/10.1016/j.asoc.2014.12.001.

[7] M. S. Kıran, H. İşcan, and M. Gündüz, "The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem," *Neural Computing and Applications*, vol. 23, no. 1, pp. 9–21, Jul. 2013, https://doi.org/10.1007/s00521-011-0794-0.

[8] D. B. Fogel, "An evolutionary approach to the traveling salesman problem," *Biological Cybernetics*, vol. 60, no. 2, pp. 139–144, Dec. 1988, https://doi.org/10.1007/BF00202901.

[9] J. McCaffrey, "Test Run - Ant Colony Optimization," *MSDN Magazine Issues*, vol. 27, no. 2, Feb. 2016, [Online]. Available: https://learn.microsoft.com/en-us/archive/msdn-magazine/2012/february/test-run-ant-colony-optimization.

[10] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, Nov. 1991, https://doi.org/10.1287/ijoc.3.4.376.

[11] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, Oct. 2007, https://doi.org/10.1007/s10898-007-9149-x.

[12] S. ArunKumar, B. V. Kumar, and M. Pandi, "Artificial bee colony optimization based energy-efficient wireless network interface selection for industrial mobile devices," *Computer Communications*, vol. 154, pp. 1–10, Mar. 2020, https://doi.org/10.1016/j.comcom.2020.01.067.

[13] H. Gao, Z. Fu, C.-M. Pun, J. Zhang, and S. Kwong, "An Efficient Artificial Bee Colony Algorithm With an Improved Linkage Identification Method," *IEEE Transactions on Cybernetics*, vol. 52, no. 6, pp. 4400–4414, Jun. 2022, https://doi.org/10.1109/TCYB.2020.3026716.

[14] S. S. Choong, L.-P. Wong, and C. P. Lim, "An artificial bee colony algorithm with a Modified Choice Function for the traveling salesman problem," *Swarm and Evolutionary Computation*, vol. 44, pp. 622–635, Feb. 2019, https://doi.org/10.1016/j.swevo.2018.08.004.