Journal of Universal Computer Science, vol. 27, no. 6 (2021), 635-645
submitted: 27/2/2021, accepted: 31/5/2021, appeared: 28/6/2021 CC BY-ND 4.0

15-Puzzle Problem Solving with the Artificial Bee Colony Algorithm Based on Pattern Database

Adem Tuncer
(Yalova University, Yalova, Turkey
https://orcid.org/0000-0001-7305-1886, adem.tuncer@yalova.edu.tr)

Abstract: The N-puzzle problem is one of the most classical problems in mathematics. Since the number of states in the N-puzzle is equal to the factorial of the number of tiles, traditional algorithms can only provide solutions for small-scale ones, such as 8-puzzle. Various uninformed and informed search algorithms have been applied to solve the N-puzzle, and their performances have been evaluated. Apart from traditional methods, artificial intelligence algorithms are also used for solutions. This paper introduces a new approach based on a meta-heuristic algorithm with a solving of the 15-puzzle problem. Generally, only Manhattan distance is used as the heuristic function, while in this study, a linear conflict function is used to increase the effectiveness of the heuristic function. Besides, the puzzle was divided into subsets named pattern database, and solutions were obtained for the subsets separately with the artificial bee colony (ABC) algorithm. The proposed approach reveals that the ABC algorithm is very successful in solving the 15-puzzle problem.

Keywords: 15-puzzle, Artificial bee colony, Pattern database, Iterative deepening
Categories: I.2., I.2.0, I.2.1, I.2.8
DOI: 10.3897/jucs.65202

1 Introduction

The N-puzzle (e.g., 8-puzzle, 15-puzzle) has been usually used to test search techniques because they can represent a particular area of problems [Drogoul and Dubreuil, 1993]. Research for the N-puzzle or sliding puzzle problem has generally focused on using traditional search methods such as the breadth-first search and deriving various heuristic methods for informed search methods such as the A-Star (A*) algorithm. The N-puzzle consists of a square board with numbered tiles: e.g., a 3×3 board for the 8-puzzle, a 4×4 board for the 15-puzzle, and a blank cell for sliding the tiles. In the puzzle, a tile is slide to a blank cell at each step to reach a goal state from the initial state and iteratively continued until the goal state is reached. The main purpose is to arrange the tiles according to the goal using as few moves as possible. Tiles can only be moved horizontally or vertically into blank cells.

Solving these sliding puzzles manually or automatically is so challenging. The state-space for the 8-puzzle contains over 10^5 nodes, the 15-puzzle space about 10^{13} nodes, and the 24-puzzle almost 10^{25} nodes [Korf and Taylor, 1996], [Piltaver et al. 2012]. The uninformed search algorithms specifically used for 8-puzzle do not require any domain-specific knowledge. On the other hand, one restriction of uninformed search algorithms is that they expand and store all possible intermediate states [Wang

and Li, 2017]. Considering the search area with trillion possible states, more than uninformed (brute force, blind) searches are required.

Figure 1 shows an example with an arbitrary initial and goal state of 15-puzzle denoting the tile by t_i and the blank tile by t_0, $<t_{11}, t_{15}, t_1, t_4, t_3, t_0, t_2, t_6, t_{10}, t_{13}, t_9, t_7, t_8, t_5, t_4, t_{12}>$ for initial state and $<t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_0,>$ for goal state shown in the figure.

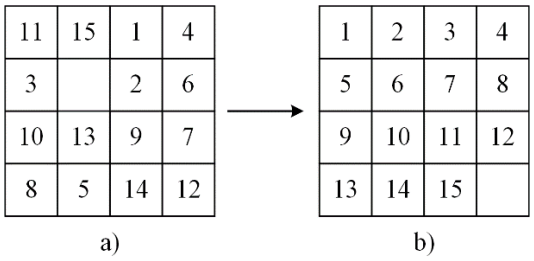

Figure 1: 15-puzzle a) arbitrary initial state, b) goal state

Some studies that use heuristic search algorithms such as A* [Hart et al. 1968] and IDA* (Iterative Deepening A*) find optimal solutions to this problem. Solution approaches to the N-puzzle problem are generally presented using different heuristic functions with the A* algorithm in the literature. Many of the implementations use the number of tiles that are not in place (misplaced tiles) or the distance of the tiles from their current location to where they should be. Manhattan Distance (MD) is used for distance measurement and is the sum of the absolute vertical and horizontal distance between the current tile and the same tile in the goal location. A* and IDA* are always capable of solving the N-puzzle problem when using an admissible heuristic function, but time required increases exponentially when the N value rises [Drogoul and Dubreuil, 1993]. Moreover, as the size of the puzzle increases, this makes it difficult to solve with the A* algorithm.

[Reinefeld, 1993] indicated that the 8-puzzle can be solved in a maximum of 31 moves using IDA*. [Nayak, 2014] proposed a solution to the 8-puzzle problem using A* and IDA* with various heuristic functions, aiming to minimize the memory required. [Korf and Schultze, 2005] provided the first complete breadth-first search for 15-puzzle. 8-puzzle is easy to solve with A* because of its small search space, but a 15-puzzle is cannot [Kendall et al. 2008]. [Korf, 1985] found the optimal solution to this problem using IDA*, eliminating the difficulties of the basic A* algorithm. [Bauer, 1994] used IDA* for the 15-puzzle with two heuristics combining Manhattan and pair distance. [Piltaver et al. 2012] examined the effect of various features of the search tree and the heuristic evaluation function on pathology and gain in the 8-puzzle. [Wang and Li, 2017] proposed a DSolving algorithm that mainly imitates the directly solving approach. The algorithm quickly moves them to their goal locations along the shortest paths for different sizes of puzzles. The authors indicated that the most distinctive feature of DSolving is that a sliding puzzle does not need to store intermediate states passed.

Some studies proposed a pattern database heuristic based on dividing the problem into sub-problems and producing solutions separately. [Korf and Felner, 2002]

presented an optimal solution to the 15-puzzle problem using IDA* with pattern database heuristics. Another study using the pattern database was carried out by [Felner and Adler, 2005] for 24-puzzle.

When examined the solution studies with meta-heuristic methods, it is seen that there are genetic algorithm-based studies for the 8-puzzle only [Shaban et al. 2010], [Igwe et al. 2013]. However, the solutions of the 8-puzzle problem with the genetic algorithm have not sufficiently been researched, and the experimental results have not fully been revealed. A similar study to the N-puzzle problem was carried out by [Naz et al. 2016]. In the study, the word tile puzzle was tried to be solved by moving a tile up, down, right, or left to find the maximum number of words using the Bee Colony Algorithm. The word can be two letters or longer. The complexity of the word tile problem is less than the complexity of the N-puzzle problem due to the variable number of words and the increased probability of the goal location of the letters for each different word. Although the studies in the literature are limited, they have generally been carried out with A* and its different versions. In this study, a meta-heuristic algorithm approach has been proposed to the solution for the 15-puzzle problem, and the artificial bee colony (ABC) algorithm, which is one of these approaches has been used. The main contributions of this study are summarized as follows:

- To the best of our knowledge, although meta-heuristic methods such as genetic algorithms are utilized for the 8-puzzle in the literature, studies for the 15-puzzle do not exist.
- The use of the ABC algorithm in the solution of this problem is the first in the literature.
- The study shows that meta-heuristic methods are also successful in solving 15-puzzle.
- The algorithm's success in finding the solution has been increased by combining the Pattern database, Linear conflict, and Manhattan distance.

2 Methodology

2.1 Principles of the ABC Algorithm

The ABC algorithm [Karaboga, 2005], one of the meta-heuristic optimization algorithms, simulates the intelligent foraging behavior of honey bee swarms. Although it has only a few control parameters when compared with other algorithms, it has been successfully applied to solving different kinds of optimization problems. In the ABC algorithm, the bees are grouped into three types, employed bees, onlooker bees, and scout bees. While the first half of the colony consists of employed bees, the second half consists of the onlooker bees. The employed bees are responsible for searching the food source around the hive and evaluate its quality. They then share the information of the food source with onlooker bees which wait in the dance area. The onlooker bees choose good food sources based on this information. The choice probability of food sources is directly proportional to their quality. The food source with better quality has a bigger chance of being selected by the onlooker bees [Karaboga and Akay, 2009]. If the nectar in a food source is depleted, the bee of that source becomes a scout bee, and the scout bee searches for a new food source randomly. In the ABC algorithm, the position of a food source corresponds to a candidate solution to the problem, and the nectar amount

of a food source represents the quality, known as fitness, of the related solution [Ozger et al. 2019]. The ABC generates a random population as initial of *SN* food sources by Eq. (1) at the first step.

$$x_{ij} = x_j^{min} + rand(0,1)\left(x_j^{max} - x_j^{min}\right) \quad (1)$$

where $i \in \{1,2,\ldots,SN\}$, $j \in \{1,2,\ldots,D\}$ and *rand(0,1)* generates a uniformly distributed random number in the range of (0,1). x_i which is a *D*-dimensional vector represents *i*th candidate solution. x_j^{max} and x_j^{min} refer lower and upper bounds of the *j*th variable, respectively.

Then, the fitness value of each solution x_i represented by fit_i is calculated by Eq. (2).

$$fit_i = \frac{1}{1 + f_i} \quad (2)$$

where f_i represents the objective function value of the solution x_i.

The employed bee searches for a new food source, v_{ij}, around the current source by Eq. (3).

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}) \quad (3)$$

where $k \in \{1,2,\ldots,SN\}$. k is a randomly selected candidate solution ($i \neq j$), and $\emptyset_{ij}$ is a random real number in the range of [-1,1]. Then, the objective function value of the source v_i is calculated and compared with the source x_i. If the objective function value of v_i is better than that of x_i, v_i is replaced with x_i, otherwise, x_i is retained. The greedy selection process is usually used for comparison.

In the standard ABC algorithm, only one parameter of each source of x_i (randomly chosen parameter, *j*) is modified. This section has been modified with an approach proposed by [Karaboğa & Akay, 2011], so the performance of the algorithm has been increased. In the modified ABC algorithm shown in Eq. (4), there is a probability that each *j* parameter changes according to a predetermined control parameter, *C*. Thus, it is possible to change more than one parameter in a food source. If no parameter changes for all *j* values, change a random parameter in the solution by Eq. (3). Thus, it is guaranteed to change at least one parameter in no case parameter is changed depending on the control parameter.

$$v_{ij} = \begin{cases} x_{ij} + \emptyset_{ij}\left(x_{ij} - x_{kj}\right), & if R_j < C \\ x_{ij}, & otherwise \end{cases} \quad (4)$$

where R_j is a uniformly distributed random number in the range of [0,1] and *C* is a control parameter in the range of [0,1] which controls the number of parameters to be modified of the ABC algorithm.

After the employed bee phase is completed, the onlooker bee phase starts. The onlooker bee chooses a food source depending on the probability value, p_i, related to its nectar amount, using Eq. (5).

$$p_{ij} = \frac{fit_i}{\sum_{j=1}^{SN} fit_i} \tag{5}$$

As with the employed bee, the onlooker bee also searches for a new food source around the existing food source. The objective function value of the new source is calculated, a greedy selection then is applied to choose between an existing source and a new source.

A parameter named limit value is defined to indicate that the food source can no longer be updated after a predetermined number of cycles for abandonment. This value shows how many times the food source visited has not been improved. If the food source reaches the *limit* value, this food source is abandoned, and then the scout bee searches for a random new food source using the Eq. (1). In the standard ABC algorithm, only one scout bee is allowed per cycle. These steps in the algorithm are repeated until a predetermined maximum number of cycles (MCN). Detailed pseudocode of the modified ABC algorithm [Karaboga and Akay, 2009] is given in the Algorithm 1.

Algorithm 1. Fundamental steps of the modified ABC algorithm

Initialize the population of solutions x_{ij}, $i = 1,2,...,SN$, $j = 1,2,...,D$
Evaluate the fitness fit_i of the population
set cycle to 1
Repeat
  For each employed bee **{**
    Produce new solution v_i by (4) and calculate fit_i
    Apply the greedy selection between x_i and v_i
  }
  For each onlooker bee **{**
    Calculate the probability values p_i for the solution x_i by (5)
    Select a solution x_i depending on p_i
    Produce new solution v_i and calculate fit_i
    Apply the greedy selection process between x_i and v_i
  }
  If there is an abandoned solution for the scout
    Then replace it with a new produced solution by (1)
  Memorize the best solution obtained so far
  cycle = cycle + 1
Until cycle = *MCN*

2.2 Solving 15-Puzzle Problem

Each food source in the ABC algorithm has a vector of directions for sliding actions labeled with '1', '2', '3', and '4' representing a move of the blank one place left, right, up, down, respectively. The standard ABC algorithm has been developed to solve continuous optimization problems. For combinatorial problems, it is necessary to convert the real numbers to integers. In this regard, Eq. (1) has been updated to generate integer numbers. A food source example is shown in Figure 2. The initial population of

candidate solutions (foods) is generally generated randomly in the standard ABC algorithm. But some of the randomly generated candidate solutions may include infeasible actions for the puzzle problem like the movements that are opposite each other are likely to be generated back-to-back. For example, a down movement after an up movement means that no action will be taken. The initial population might be generated by preventing back-to-back opposite movements to overcome this problem. However, the same problem might come up in the new food sources that will be produced from the old food sources. In this case, it may not be sufficient to produce a feasible initial population. Furthermore, initialization with feasible solutions requires more time. This process may reduce the search capability of the algorithm and may increase the time to find an optimal solution [Karaboga and Akay, 2011].

Figure 2: Example of the food source in the ABC algorithm

It is obvious that generation candidate solutions considering the constraints will add an additional cost to the algorithm. Therefore, candidate solutions were generated without considering the constraints in the study. The movements that do not comply with the constraints in the food sources have been ignored and removed from the food source. Thus, a constrained optimization problem has been replaced by an unconstrained optimization problem ignoring the constraint with movements. Examples of possible constraints for movements in candidate solutions are shown in Figure 3.

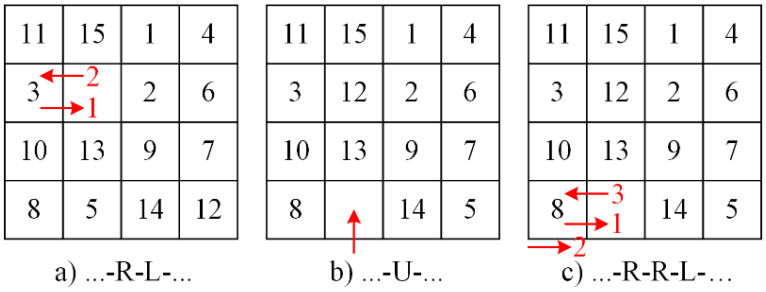

Figure 3: Constraints. a) back-to-back opposite moves, b) moves that exceed the boundary of the puzzle, c) moves that exceed the boundary of the puzzle between back-to-back opposite moves

The heuristic function based on MD used in the study has been enhanced by linear conflict (LC). The linear conflict heuristic was the first described in [Hansson et al. 1992]. Two tiles t_i and t_j are in a linear conflict if they are in the same row or column, their goal positions are both in that row or column, both tiles are different positions than they should be. MD and LC calculations are shown in Eqs. (6) and (7), respectively.

For tiles with conflict, the cost of linear conflict heuristics will be at least 2 more than MD. The objective function used as the sum of the MD and LC is shown in Eq. (8).

$$MD = |x_1 - x_2| + |y_1 - y_2| \tag{6}$$

$$LC = \begin{cases} 1, & linear\ conflict\ exists \\ 0, & otherwise \end{cases} \tag{7}$$

$$f_i = \sum_{i=1}^{N-1} MD(t_i) + 2 \times LC(t_i) \tag{8}$$

where N represents the size of the puzzle and t_i is the i. tile. In the row or column conflict, extra movement is required for the movement of the tiles and this cost is added to the MD.

2.3 Pattern Database

A pattern is the partial specification of an N-puzzle and a method [Culberson and Schaeffer, 1998] based on dividing the N-puzzle into subsets and solving each subset separately, ignoring the other tiles. This database has been referred to as the fringe. Studies [Korf and Felner, 2002], [Felner and Adler, 2005], [Culberson and Schaeffer, 1998] show that it is more effective to solve the puzzle by dividing it into subsets instead of solving the entire puzzle.

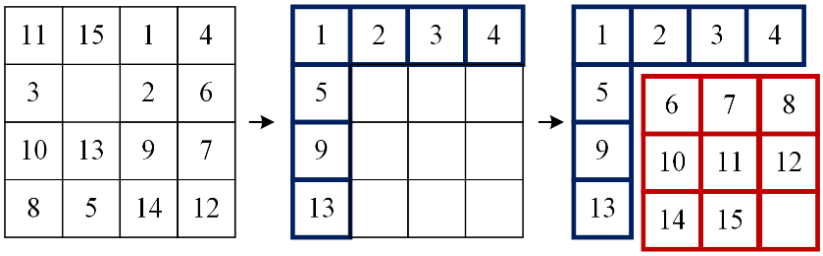

Figure 4: The fringe pattern for the 15-puzzle

Figure 4 shows a 15-puzzle divided into two subsets. As can be seen in the figure, the tiles of <t1, t2, t3, t4, t5, t9, t13> are placed where they should be, and the locations of the other tiles are ignored. Then the rest tiles are located. In the study, a pattern database has been used as in the figure to increase the ABC algorithm's effectiveness in solving the puzzle.

The efficiency of the ABC algorithm's search strategy has been increased by including an iterative deepening search approach. The iterative deepening search, which is of the uninformed search strategies, is a method with an increasing depth limit until a solution is found. In the study, the steps in the ABC algorithm to approach the solution iteratively have been updated with inspiration from the iterative deepening search. The algorithm firstly tries to find a solution by generating food resources

according to the predetermined number of moves within a depth of 1. Only if it does not find the goal state, the algorithm then initializes the food sources and starts searching within a depth of 2. The value of depth is incremented iteratively until the predefined depth value. The initial state of the puzzle at each depth is the best state obtained at the previous depth. Thus, the algorithm starts a little closer to the solution at each depth and the time to reach the solution is shortened. This approach improves the heuristic search with the strength of iterative deepening.

3 Experiments

In the study, the ABC algorithm was applied by combining the advantages of Manhattan distance and linear conflict heuristics, and pattern database. The algorithm was run on 25 randomly generated solvable instances of the 15-Puzzle. The parameters of the ABC algorithm are as follows: the number of foods is 80, the limit number is 100, and the max cycle is 100 for each depth. The food length is determined dynamically, different at each depth. As the goal is approached more at each depth, the food length, which is initially determined as large, is gradually reduced. In this study, food length is taken as an initial value of 30. Although a fixed-length value for food length has been determined, this value becomes less in terms of the total number of movements when consecutive undesirable moves in the sources are ignored.

Table 1 summarizes the results, averaged over 10 independent runs varying the initial states of the puzzles. In the table, the best, mean, and worst number of moves obtained using the ABC algorithm are given. In addition, the number of moves in the solution of each instance with the IDA* has been also added to the table. However, with IDA*, it is seen that the number of moves in solutions is less. The reason for this is that the approach has a deterministic feature. The number of moves in the meta-heuristic algorithm is greater than IDA* but the difference is negligible. Heuristic algorithms are expected to produce acceptable results in a reasonable time, rather than always the optimum results by nature. Considering the number of moves in the experimental results, it is clearly seen that the ABC algorithm produces acceptable results.

In Figure 5, the convergence graphic of the ABC algorithm running based on iterative deepening is given for the first instance in the Table 1. Convergence graphs also for the other instances occur similarly. As can be seen from the figure, there are two distinct convergences. The first convergence graphic shows the convergence in the solution of the first subset of the puzzle divided into two subsets, and the other graphic shows the convergence in the solution of the second. It is seen from the graphic; the first subset of the puzzle was solved at the 3rd depth and the second at the 2nd depth.

No	Initial state	Number of moves			
		IDA*	ABC		
			Best	Mean	Worst
1	1 5 2 7 10 14 11 6 15 12 9 3 13 0 8 4	34	37	42	45
2	5 6 10 7 1 3 11 8 13 4 15 9 14 0 2 12	38	43	47	51
3	1 11 6 2 10 13 15 5 3 12 0 4 9 7 14 8	40	46	54	62
4	6 5 2 7 13 0 10 12 4 1 3 14 9 11 15 8	44	49	59	68
5	4 3 10 7 6 0 1 2 12 15 5 14 9 13 8 11	44	52	60	68
6	4 10 3 2 1 0 7 8 9 6 13 15 14 12 11 5	44	51	58	69
7	3 4 11 2 9 1 14 15 7 6 0 8 5 13 12 10	44	51	60	72
8	3 10 2 5 15 6 13 4 0 11 1 7 9 12 8 14	46	52	68	79
9	9 4 0 3 14 7 5 12 15 2 13 6 10 1 8 11	46	54	66	79
10	7 1 12 10 6 11 15 4 0 2 5 14 3 13 8 9	48	59	71	85
11	1 13 5 7 14 9 10 12 11 8 2 15 6 0 4 3	48	62	78	88
12	13 9 5 12 10 2 4 11 3 8 0 7 1 14 6 15	48	64	77	86
13	2 13 6 1 14 5 11 0 12 4 8 10 9 3 15 7	50	66	77	89
14	11 3 12 9 2 8 10 14 0 7 15 13 1 6 5 4	50	68	78	92
15	7 6 15 12 14 1 13 3 0 9 8 4 2 11 5 10	50	68	81	94
16	5 8 13 15 14 0 1 7 4 6 10 2 11 9 12 3	52	59	77	92
17	12 2 5 11 10 0 1 6 3 14 8 9 7 4 13 15	52	62	78	91
18	13 3 2 8 12 0 5 1 11 6 9 15 4 14 7 10	52	63	76	89
19	7 13 1 4 9 12 8 5 15 14 0 6 11 2 3 10	52	59	78	91
20	8 11 12 10 2 0 15 1 14 6 4 3 7 9 5 13	54	61	81	93
21	6 8 12 13 7 2 5 14 9 3 1 15 11 0 10 4	54	65	77	94
22	9 12 2 5 11 1 10 14 0 4 3 8 6 15 7 13	54	67	80	92
23	10 12 11 7 8 9 14 5 3 13 4 1 6 0 2 15	56	69	80	96
24	3 10 14 5 1 12 11 8 15 7 9 6 2 0 13 4	56	71	81	95
25	9 3 12 5 4 14 6 11 8 7 15 13 10 0 2 1	56	71	81	92

Table 1: Solutions for 25 randomly generated 15-Puzzle instances using ABC algorithm

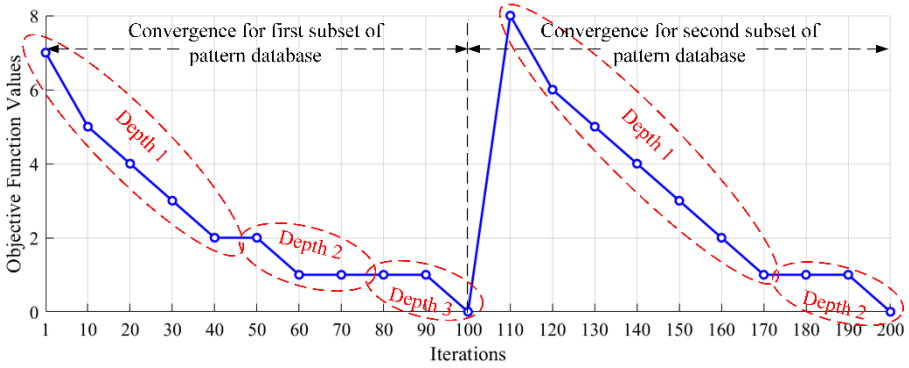

Figure 5: Convergence graphic for ABC algorithm based on iterative deepening and pattern database

4 Conclusion

A new approach based on the meta-heuristic method using the ABC algorithm with a solving of the 15-puzzle problem has been proposed in the study. The efficiency of the Manhattan distance heuristic was improved by combining it with the linear conflict function. The pattern database method was implemented for reaching the solution faster. Besides, the efficiency of the ABC algorithm's search strategy was increased by including an iterative deepening search approach. It is clear that the ABC algorithm produces acceptable results given the number of moves in the experimental results, which proves that the ABC algorithm is an effective method for solving the 15-puzzle problem. It is obvious that the problem cannot be solved with traditional algorithms as the dimensions of the N-puzzle grow. In this respect, meta-heuristic algorithms such as the ABC algorithm can be used effectively in the solution of the N-puzzle problem.

References

[Bauer, 1994] Bauer, B: The manhattan pair distance heuristic for the 15-puzzle. Paderborn, Germany, 1994

[Culberson and Schaeffer, 1998] Culberson, J.C., Schaeffer, J.: Pattern databases. Computational Intelligence, 14(3), 318-334, 1998

[Drogoul and Dubreuil, 1993] Drogoul, A., Dubreuil, C.: A distributed approach to n-puzzle solving. In: Proceedings of the Distributed Artificial Intelligence Workshop. Citeseer, 1993

[Felner and Adler, 2005] Felner, A., Adler, A.: Solving the 24 puzzle with instance dependent pattern databases. In: International Symposium on Abstraction, Reformulation, and Approximation, 248-260. Springer, 2005

[Hansson et al. 1992] Hansson, O., Mayer, A., Yung, M.: Criticizing solutions to relaxed models yields powerful admissible heuristics. Information Sciences, 63(3), 207-227, 1992

[Hart et al. 1968] Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, 4(2), 100-107, 1968

[Igwe et al. 2013] Igwe, K., Pillay, N., Rae, C.: Solving the 8-puzzle problem using genetic programming. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, 64-67, 2013

[Karaboga, 2005] Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Tech. rep., Citeseer, 2005

[Karaboga and Akay, 2009] Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Applied mathematics and computation, 214(1), 108-132, 2009

[Karaboga and Akay, 2011] Karaboga, D., Akay, B.: A modified artificial bee colony (abc) algorithm for constrained optimization problems. Applied soft computing, 11(3), 3021-3031, 2011

[Kendall et al. 2008] Kendall, G., Parkes, A., Spoerer, K.: A survey of np-complete puzzles. ICGA Journal, 31(1), 13-34, 2008

[Korf, 1985] Korf, R.E.: Depth-first iterative-deepening: An optimal admissible tree search. Artificial intelligence, 27(1), 97-109, 1985

[Korf and Felner, 2002] Korf, R.E., Felner, A.: Disjoint pattern database heuristics. Artificial intelligence, 134(1-2), 9-22, 2002

[Korf and Schultze, 2005] Korf, R.E., Schultze, P.: Large-scale parallel breadth-first search. In: AAAI, vol. 5, 1380-1385, 2005

[Korf and Taylor, 1996] Korf, R.E., Taylor, L.A.: Finding optimal solutions to the twenty-four puzzle. In: Proceedings of the national conference on artificial intelligence, 1202-1207. Citeseer, 1996

[Nayak, 2014] Nayak, D.: Analysis and implementation of admissible heuristics in 8 puzzle problem. Ph.D. thesis 2014

[Naz et al. 2016] Naz, E., Al-Dabbas, K., Abrishami, M., Mehnen, L., Cvetkovic, M.: Solving Word Tile Puzzle using Bee Colony Algorithm. International Journal of Advanced Computer Science and Applications, 7(11), 229-234, 2016

[Ozger et al. 2019] Ozger, Z.B., Bolat, B., Diri, B.: A Probabilistic Multi-Objective Artificial Bee Colony Algorithm for Gene Selection. Journal of Universal Computer Science, 25(4), 418-443, 2019

[Piltaver et al. 2012] Piltaver, R., Luštrek, M., Gams, M.: The pathology of heuristic search in the 8-puzzle. Journal of Experimental & Theoretical Artificial Intelligence, 24(1), 65-94, 2012

[Reinefeld, 1993] Reinefeld, A.: Complete solution of the eight-puzzle and the benefit of node ordering in ida*. In: International Joint Conference on Artificial Intelligence, 248-253. Citeseer 1993

[Shaban et al. 2010] Shaban, R.Z., Natheer Alkallak, I., Mohamad Sulaiman, M.: Genetic Algorithm to Solve Sliding Tile 8-Puzzle Problem. Journal of Education and Science, 23(3), 145-157, 2010

[Wang and Li, 2017] Wang, G., Li, R.: Dsolving: a novel and efficient intelligent algorithm for large-scale sliding puzzles. Journal of Experimental & Theoretical Artificial Intelligence, 29(4), 809-822, 2017