A Genetic Algorithm for Solving Beehive Hidato Puzzles

Matheus Müller Pereira da Silva and Camila Silva de Magalhães

Universidade Federal do Rio de Janeiro - UFRJ, Campus Xerém,
Duque de Caxias, RJ 25245-390, Brazil
mullerpds@ufrj.br,camila@xerem.ufrj.br

Abstract. Beehive Hidato puzzles are logic games, similar to Sudoku, in which the grid cells are hexagons. Some hexagons are prefilled with given numbers, and the objective of the game is to find a path of natural numbers, from 1 to the grid size n, in such a way that consecutive numbers stay connected by any hexagon side. Although the rules of the game are simple, finding the solution to this problem can be quite challenging. In this work, we designed and implemented a genetic algorithm (GA) to solve Beehive Hidato problems. The proposed GA uses common genetic operators and the RTS niching technique to preserve population diversity. A new strategy based on gene convergence rate is also implemented and tested. The proposed algorithm was evaluated in 21 instances of Beehive Hidato with different sizes and complexities. The results show that GAs are promising tools for solving Beehive Hidato problems.

Keywords: Genetic Algorithm, Hidato Puzzles, Niching

1 Introduction

Hidato (from the Hebrew *hida* that means riddle) is a puzzle logic game, similar to the very popular Sudoku. This game was proposed by the Israeli mathematician Gyora M. Benedek [9]. The Hidato puzzle consists of n grid cells in which the player has to complete the cells with natural numbers from 1 to n, in such a way that consecutive numbers stay in adjacent cells. Some of the Hidato grid cells are fulfilled with given numbers that are unchanged, including always the first and the largest number (1 and n). While the Hidato rules are very simple, completing the game grid can be quite challenging. The Beehive Hidato is a variant of the classic squared grid cell Hidato and was inspired by honeycomb structures. The grid cells of Beehive Hidato adopt the hexagonal format, and consecutive numbers can be placed on any hexagon connected with other through any of its sides. Hidato games have different difficulty levels related to the size of the Hidato grid (number of hexagons) and also the number of fixed given numbers. The larger the number of hexagons and the smaller the amount of fixed given numbers, the more difficult the Hidato problem is. The hexagonal grid can also contain holes, which means that some places (hexagonal cells) are not allowed to be filled with any number.

Although a lot of research effort has been made to solve the Sudoku game using many different techniques [5, 6, 11–13] to the best of the author's knowledge, this is the first paper to address the Hidato problem. Most of the works for solving Sudoku uses nature inspired optimization techniques, such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA), with relatively good performance. In this work, we propose a genetic algorithm [4] specifically designed to solve Beehive Hidato puzzles. We are interested in investigating the capability of genetic algorithms in solving the Beehive Hidato problem. The proposed GA uses a steady-state population model and commonly used crossover and mutation operators for permutation representation problems. The population diversity is maintained by using the Restricted Tournament Selection (RTS) [1] niching technique with a dynamic tournament size (w) [10]. A new convergence based gene insertion strategy is also proposed and tested in an attempt to improve performance and convergence speed.

The proposed GA was applied to 21 instances of the Beehive Hidato problem of different sizes and complexities. The results show that the proposed GA can easily handle small size Beehive Hidato problems. Nevertheless, for the class 2 hidato instances tested, the success rate varies from 3% to 100% depending on the Hidato complexity.

The rest of the paper is organized as follows. Section 2 describes the Beehive Hidato rules and the new proposed GA. Section 3 presents the results obtained with the application of the GA proposed to the 21 Beehive Hidato instances and analyses the influence of the RTS technique on it. The conclusions of the work done are presented in Section 4.

2 Beehive Hidato Rules and The Proposed Genetic Algorithm

2.1 Hidato Rules

In Beehive Hidato, initially, there is a grid of hexagons with some prefilled fixed numbers (Fig. 1a). The aim is to fill the empty hexagons with consecutive numbers from 1 to the largest number in the grid. In other words, 1 must be adjacent to 2, 2 must be adjacent to 3 and so on, generating a continuous path. Every Hidato puzzle has a unique solution. Fig. 1b shows the complete solution and its corresponding path for this Hidato.

Beehive Hidato, similar to Sudoku, is a constraint satisfaction fill-in puzzle with simple rules [7].

- Rule 1: Each hexagonal cell of the grid must be filled with a natural number ranging from 1 to n;
- Rule 2: Each number, ranging from 1 to n, is only in one hexagonal cell (i.e., number repetition is not allowed on grid)
- Rule 3: Prefilled numbers are not allowed to change.

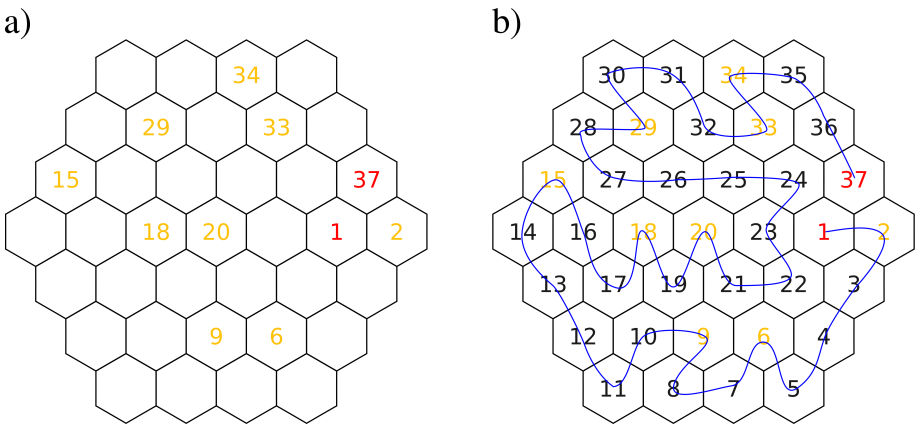

Fig. 1. Initial hidato grid with prefilled fixed numbers (*yellow*), first and last numbers (*red*) (a). Complete solution and its corresponding path (*blue line*) (b).

2.2 The Proposed GA

The proposed method is a steady-state genetic algorithm specifically designed to solve hexagonal Hidato puzzles. In a steady-state GA, a newly created offspring is immediately tested for insertion in the population. The GA initiates generating a random initial population of candidate Beehive Hidato solutions. Parents individuals are randomly selected, and the genetic operators are applied (crossover followed by mutation). The offspring are inserted in the population according to the Restricted Tournament Selection (RTS) method [1]. The population evolves until the maximum number of fitness function evaluations (FEs) allowed or until the global optimum is found.

Representation Each Beehive Hidato solution is represented by a permutation of integers of 1 to n, where n is the total number of genes on a given chromosome (e.g.: {1, 5, 3, 2, 4}). Thus, number repetitions are not allowed. In this representation, the numbers of the Hidato are organized linearly, always starting from the top left to the bottom right hexagonal cell of the grid. However, fixed prefilled numbers of Hidato are not included in the chromosome. These given numbers are removed from the initial population generation (randomly arranging only the non fixed numbers of the problem) and are considered only in order to complete the solution in the fitness evaluations.

Fitness Function A fitness function was specially designed to quantify the quality of a solution for the Beehive Hidato problem, considering the rules of the puzzle. Solutions of Beehive Hidato comprises hexagonal cells filled with consecutive numbers linked adjacently. So, for each hexagonal cell of the Beehive Hidato, we sum one point to each following number (successor or predecessor)

that is connected to it in an adjacent hexagonal cell. So, for a Beehive Hidato of size N, the fitness of the optimum solution is equal to $(N * 2) - 2$. Two points are subtracted from $N * 2$ to take into account the first and last numbers, which have only one next number to it. In other words, the fitness function checks for every number n, if there are consecutive numbers ($n + 1$ and $n - 1$) adjacent to it, assigning one point to each of these conditions when true. The proposed GA was designed to maximize this fitness function.

A pseudo-code for evaluating a Beehive Hidato solution can be seen below:

```
fitness_score = 0
For each number n in candidate solution:
    For every adjacent number n_a of n:
        If n_a equals to n + 1:
            fitness_score = fitness_score + 1
        If n_a equals to n - 1:
            fitness_score = fitness_score + 1
```

Diversity Preservation To promote population diversity and, therefore, avoid premature convergence to local optimum, the RTS crowding technique [1] was implemented. This technique is based on the concept of local competition in which similar individuals competes to stay in the population. In this scheme, parents are randomly selected and a newly generated individual is inserted in the population replacing the most similar among w randomly selected individuals if it is better than the most similar one. The distance between two solutions for the Beehive Hidato problem was defined here as the number of hexagonal cells with distinct values. For example, given $chromosome_1 = \{1, 5, 3, 2, 4\}$ and $chromosome_2 = \{1, 5, 2, 4, 3\}$, the distance between them is evaluated as 3.

In this AG, the w parameter was dynamically set to linearly decay from 100% of the population to 10%, during whole execution [10]. In this way, strong niche formation is favored at the beginning of the evolution, and it is relaxed towards the end.

Genetic Operators The genetic operators used were two commonly used operators for permutation representations in GAs. Only one crossover and one mutation operator were used. The crossover operator used was the Partially Mapped Crossover (PMX) as described by Whitley [2]. The mutation operator used was swap mutation, as described in [3]. This operator works by randomly choosing two genes and swapping their values. Crossover probability was set to 90%, and mutation probability was set to 10%.

Convergence Based Gene Insertion - (CBGI) Strategy In an attempt of to improve the algorithm performance in finding the correct order of numbers in the hexagonal grid even when the Beehive Hidato problem has few fixed given numbers, we implemented a convergence based gene insertion (CBGI) strategy.

This strategy works as follows: after a elapsed evolution time S (a percentage of the total evolution time, calculated on the maximum number of fitness function evaluations), the algorithm evaluates for each gene of the best individual its allele frequency, considering all individuals in the GA population. This process occurs periodically (in *epoch*s). If the allele frequency of a gene of the best individual in the population is higher than a parameter c, the value of the gene is inserted as a given fixed number for the Beehive Hidato problem, and it is held in that way until the end of the evolution.

More specifically, for each gene of the current best individual, the algorithm evaluates the number of occurrences of its gene value, in the same position, for all individuals in the population. Then, this quantity is divided by the population size, to get its rate. The gene with the highest convergence rate (above of a parameter c) is then selected to be inserted in the Beehive Hidato problem as a fixed number, in its corresponding position.

3 Results

The performance of the steady-state GA implemented was evaluated for 21 hexagonal Beehive Hidato problems. These problems were randomly selected from two different databases [8, 9] and used as initial test problems for the algorithm's performance. From those, problems 1 to 10 are composed by 19 hexagons (class 1 problems), problems 11 to 21 have 37 hexagons (class 2 problems). Therefore, the fitness value of the globally optimum solution for class 1 and class 2 problems is 36 and 72, respectively.

All experiments were performed using a population size of 250 individuals and a maximum of $2.0E+05$ fitness function evaluations. For all problem instances, 30 independent GA runs were performed.

The parameters related to the convergence based gene insertion (GBGI) strategy used were: S set to 20%, *epoch* to 5% and c to 20%. It means that, after 20% of the total maximum number of fitness function evaluations, and then on each 5% of the evolution, a gene is inserted as part of the Beehive Hidato problems if its rate of occurrence in the population surpass c.

The algorithm performance was evaluated with and without the use of the GBGI strategy. Tables 1 to 4 shows, for every problem instance: the best fitness found, the averaged best fitness, the averaged number of fitness function evaluations to reach the best solution found and the success rate (SR). SR is defined as the number of successful runs (i.e., runs in which the global optimum solution was found) over the total number of runs.

Tables 1 and 2 show results for problems 1-10 and 11-21, respectively, without using the CBGI strategy (described in section 2.2).

Tables 3 and 4 show results for the GA when using the CBGI strategy, with parameters S and c set to 20%.

For all class 1 Beehive Hidato problem instances (problems 1 to 10, show in Tables 1 and 3), the genetic algorithm implemented can find the global optimum solution with a success rate of 100% with and without the use of the CBGI

Table 1. Beehive Hidato class 1 problems without the CBGI strategy

problem	best fit.	avg. best fit.	st. dev.	avg. FEs	st. dev.	SR
1	36	36.00	0.00	6728.07	1600.58	1.00
2	36	36.00	0.00	4626.67	1012.95	1.00
3	36	36.00	0.00	8990.13	2583.16	1.00
4	36	36.00	0.00	19549.20	13161.44	1.00
5	36	36.00	0.00	4898.13	1300.68	1.00
6	36	36.00	0.00	8771.80	2126.81	1.00
7	36	36.00	0.00	6284.13	1842.03	1.00
8	36	36.00	0.00	8004.20	1927.22	1.00
9	36	36.00	0.00	7841.07	2492.01	1.00
10	36	36.00	0.00	4458.73	1679.17	1.00
average:	36	36.0	0.0	8015.21	2972.61	1.00

Table 2. Beehive Hidato class 2 problems without the CBGI strategy

problem	best fit.	avg. best fit.	st. dev.	avg. FEs	st. dev.	SR
11	72	71.60	0.81	78498.20	23461.47	0.80
12	72	70.60	1.83	94339.00	39673.09	0.60
13	72	72.00	0.00	48389.20	6871.02	1.00
14	72	69.80	1.32	93993.13	31468.55	0.17
15	72	68.53	1.66	134916.33	28188.26	0.03
16	72	68.67	1.69	140021.33	28992.19	0.03
17	72	70.80	1.24	102235.80	30075.19	0.47
18	72	70.13	0.51	84031.67	18814.75	0.07
19	72	67.33	2.06	150680.13	28936.90	0.03
20	72	71.93	0.37	46169.40	9177.76	0.97
21	72	71.47	1.04	78580.47	15773.78	0.77
average:	72	70.26	1.14	95623.15	23766.63	0.45

strategy. These class 1 Beehive Hidato problem instances differ in the quantity and values of the fixed given numbers. The number of given numbers varies, in this case, from 6 (problem 4) to 8 (problems 2, 5 and 10). For all other problems, seven numbers are given. As expected, the averaged number of fitness evaluations to reach the global optimum is higher to the problem with less given numbers (problem 4). When using the CBGI strategy (Table 3), we observed a slightly reduction in the averaged number of fitness evaluations to reach the optimum. The main difference is found for problem 4 in which the averaged number of FE dropped off about 2500 evaluations.

Table 3. Beehive Hidato class 1 problems with the CBGI strategy

problem	best fit.	avg. best fit.	st. dev.	avg. FEs	st. dev.	SR
1	36	36.00	0.00	6914.60	1637.27	1.00
2	36	36.00	0.00	4368.67	1202.92	1.00
3	36	36.00	0.00	8176.93	2898.01	1.00
4	36	36.00	0.00	16811.07	10453.98	1.00
5	36	36.00	0.00	4754.73	1461.79	1.00
6	36	36.00	0.00	9156.93	2757.87	1.00
7	36	36.00	0.00	6343.33	1550.25	1.00
8	36	36.00	0.00	7018.67	2281.46	1.00
9	36	36.00	0.00	7864.73	2388.08	1.00
10	36	36.00	0.00	4396.40	1355.34	1.00
average	36	36.0	0.0	7580.6	2798.7	1.00

Table 4. Beehive Hidato class 2 problems with the CBGI strategy

problem	best fit.	avg. best fit.	st. dev.	avg. FEs	st. dev.	SR
11	72	71.93	0.37	78173.47	23525.59	0.97
12	72	70.13	2.03	84586.60	21264.76	0.53
13	72	71.93	0.37	47842.33	5929.11	0.97
14	72	70.00	1.49	96046.20	32206.56	0.27
15	72	69.07	1.55	145073.33	33799.58	0.07
16	72	69.27	1.23	124719.00	19956.63	0.03
17	72	70.80	1.13	87628.47	13251.76	0.43
18	72	70.27	0.69	80048.73	15193.18	0.13
19	72	68.53	2.16	138049.33	24089.08	0.13
20	72	71.47	0.90	44107.53	9508.78	0.73
21	72	71.60	0.97	74137.13	13698.76	0.83
average:	72	70.45	1.17	90946.56	19311.25	0.46

For class 2 Beehive Hidato (problems 11 to 21) the GA performance varies significantly from instance to instance (Tables 2 and 4). However, the GA implemented was able to find the global optimum solution for all problem instances tested. The fixed given number of these instances ranges from 9 (problems 15, 16 and 19) to 13 (problem 13). Problems 17 and 21 have ten fixed given numbers, problems 12, 14 and 18 have 11, and problems 11 and 20 have each one, 12 prefilled numbers.

For 5 of the 11 class 2 problem instances, the proposed algorithm presented a success rate higher or equal to 60% in finding the global optimum solution. For 2 of these, the algorithm SR is equal or higher than 80%.

As expected, for problem instances with less fixed given numbers (problems 15, 16 and 19), the algorithm presented an inferior performance with SR of only 3%. The same is observed with problem number 18, for which the SR obtained was of 7%. The SR of the algorithm for problems 14 and 17 (with 11 and 10 fixed given numbers) were of 17% and 47%, respectively.

When the CBGI strategy is used in the GA to solve class 2 Beehive Hidato problems (Table 4), no significant change in the average algorithm performance was observed. However, an improvement of 10% in SR was found for problems 19 and 14, two difficult problems. A higher and more significant improvement was observed only for problem 11, for which the SR increased to 17%. Problems 15, 18 and 21 had minor improvements (4% to 6%). For the other class 2 problem instances tested, the SR of the GA using the CBGI strategy was equal or worse than the GA results without using it. However, the average FE for reaching the global optimum is lower when the CBGI is used for all cases, except for problem 14, in which the GA-CBGI approach has a slightly higher average FE (Table 4).

The proposed GA can quickly solve class 1 Beehive Hidato problems with 100% of success rate. However, for class 2 problems the success rate of the GA proposed varies from 3% (more difficult problems, with only nine fixed given numbers) to 100%. From the 11 class 2 problem instances tested, the GA and the GA with the CBGI strategy obtained a success rate above 50% for five instances.

3.1 Influence of the Diversity Preservation Technique

To evaluate the influence of the RTS niching technique on the results, we tested the proposed GA on the 21 Beehive Hidato problems using another selection-insertion technique, without a diversity preservation mechanism. All the other GA parameters, operators, and strategies remained the same. The only modification was that the parent selection was performed by binary tournament selection [3]. Besides, a newly generated offspring in the population was inserted in the population replacing the worst individual. Results for class 1 and class 2 Beehive Hidato problems are shown in Tables 5 and 6, respectively.

For class 1 Hidato problems, although the GA can find the global optimum, we observe a decreased success rate for all ten instances tested. The most significative performance loss was observed for problem four (the most difficult class 1 problem tested). For problems one and three the success rate also decreased

from 100% (Table 3) to 57% and 47% respectively (Table 5). For the other class 1 problem instances, the success rate varied from 70% to 93%. However, the averaged best fitness value is worse than the results with the RTS niching technique in all cases.

Table 5. Beehive Hidato class 1 problems with CBGI strategy without the RTS technique

problem	best fit.	avg. best fit.	st. dev.	avg. FEs	st. dev.	SR
1	36	35.07	1.14	1818.47	518.19	0.57
2	36	35.80	0.61	1374.27	298.78	0.90
3	36	34.60	1.50	7646.53	18150.42	0.47
4	36	32.73	2.00	4191.13	7366.29	0.23
5	36	35.87	0.51	1394.20	341.57	0.93
6	36	35.53	1.14	4421.53	7575.71	0.83
7	36	35.53	1.14	1895.13	596.72	0.83
8	36	35.40	0.93	2048.67	401.97	0.70
9	36	35.00	1.72	3361.20	7127.55	0.73
10	36	35.60	1.22	2004.40	3152.68	0.87
average:	36.	35.11	1.19	3015.55	4552.99	0.71

Table 6. Beehive Hidato class 2 problems with CBGI strategy without the RTS technique

problem	best fit.	avg. best fit.	st. dev.	avg. FEs	st. dev.	SR
11	68	62.20	3.65	21983.80	36841.56	0.00
12	64	60.93	2.61	15769.87	14025.78	0.00
13	70	62.27	3.81	15763.60	24061.69	0.00
14	70	62.20	4.08	15180.93	21451.34	0.00
15	66	58.13	3.36	12808.20	16630.51	0.00
16	64	58.73	2.49	33452.67	49899.87	0.00
17	68	61.33	3.46	16482.93	21146.48	0.00
18	70	62.27	3.67	16412.40	22317.50	0.00
19	62	57.80	2.25	15717.07	17662.78	0.00
20	70	64.33	3.33	15056.60	13788.24	0.00
21	70	61.20	3.81	23244.27	27569.69	0.00
average:	67.46	61.04	3.32	18352.03	24126.86	0.0

When we compare the results with and without the use of the RTS technique for class 2 Beehive Hidato problem instances we see that the algorithm cannot find the global optimum solution for any of the 11 problems tested (Table 6). Although solutions near to the global optimum were found for five problems

(best fit. values of 70), the averaged best fit is significantly worse than the results using the RTS strategy in all cases. The averaged best fitness without the utilization of the RTS is below 65 (Table 6) whereas when using the RTS strategy it is above 68 (Table 4) for all cases. The RTS technique has a crucial role in avoiding premature convergence, preserving diversity and improving the algorithm performance significantly.

4 Conclusion

In this work, we presented a genetic algorithm for solving the Beehive Hidato Problem. To the best of the author's knowledge, this is the first work to adress this problem. A GA representation and a form to evaluate the individual fitness were specifically designed and implemented to solve the Beehive Hidato problem. The proposed algorithm uses GA operators commonly used for permutation representation problems, and the RTS niching technique, to assure diversity preservation, an important aspect in solving logic based puzzles [5]. A new strategy based on gene rate convergence (CBGI) is also proposed and tested in an attempt to improve the GA search, progressively reducing the search space by inserting new fixed given numbers to the problem during evolution. The main improvement observed with the inclusion of the CBGI strategy was: (i) a higher success rate for more challenging cases; and (ii) a higher convergence speed (lower averaged number of FEs to reach the global optimum).

Although the mean results for the proposed GA and the GA-CBGI are very similar, we believe that this strategy can be further improved to give better results. The main drawback of this approach is that once a gene value is inserted in a wrong position, it will be a part of the problem and will stay unchanged until the end of the evolution, taking the algorithm in the direction of a wrong solution. Rather than inserting genes with high convergence rate as part of the Beehive Hidato problem, a strategy that spreads the values of the best individual's genes to the chromosomes of others individuals in the population may be a better option. These analyses are in progress and will be reported soon.

References

1. Harik, G.R.: Finding Multimodal Solutions Using Restricted Tournament Selection. ICGA (1995)
2. Whitley, D.: Permutations, Handbook of Evolutionary Computation (1997)
3. Eiben, A.E., James E.S.: Introduction to Evolutionary Computing. Springer (2003)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley (1989)
5. Segura, C. et al.: The importance of diversity in the application of evolutionary algorithms to the Sudoku problem. Evolutionary Computation (CEC), IEEE Congress (2016)
6. Singh, G., Deep, K.: A new membrane algorithm using the rules of Particle Swarm Optimization incorporated within the framework of cell-like P-systems to solve Sudoku. Applied Soft Computing. 54, 27-39 (2016)

7. Bartos, S.: Effective encoding of the Hidato and Numbrix puzzles to their CNF representation. Charles University, Bachelor thesis (2014)
8. Hidoku Solver Online, `http://hidoku-solver.appspot.com/` [Accessed 15 Aug. 2017]
9. Hidato, `http://www.hidato.com/` [Accessed 15 Aug. 2017]
10. de Magalhes, C.S., Almeida, D. M., Barbosa, H. J. C., Dardenne, L.E.: A dynamic niching genetic algorithm strategy for docking highly flexible ligands. Information Sciences. 289: 206-224 (2014)
11. Mantere T., Improved ant colony genetic algorithm hybrid for sudoku solving, in Information and Communication Technologies (WICT), Third World Congress on, Dec 2013, pp. 274279 (2013)
12. Z. Wang, T. Yasuda, and K. Ohkura, An evolutionary approach to sudoku puzzles with filtered mutations, in Evolutionary Computation (CEC), 2015 IEEE Congress on, May 2015, pp. 17321737 (2015)
13. L. Clementis, Advantage of parallel simulated annealing optimization by solving sudoku puzzle, in Emergent Trends in Robotics and Intelligent Systems, ser. Advances in Intelligent Systems and Computing, P. Sink, P. Hartono, M. Virkov, J. Vak, and R. Jaka, Eds. Springer International Publishing, 2015, vol. 316, pp. 207213 (2015)