doi: 10.14355/ijepr.2014.0302.02

Solving Kakuro Puzzle using Self Adapting Harmony Search Metaheuristic Algorithm

Stojanche Panov[1], Saso Koceski[2]

Faculty of Computer Science, "Goce Delchev" University – Stip,blvd. Krste Misirkov bb., 2000 Stip, R. Macedonia

[1]stojanche.panov@ugd.edu.mk; [2]saso.koceski@ugd.edu.mk

Received 26th Feb. 2013; Accepted 6th Aug. 2013; Published 24th April 2014

Abstract

In paper presents a Self Adapting Harmony Search (SAHS) metahauristic based solution to the grid-based Japaneese game know as Kakuro. It has been shown that our approach can provide a time-efficient solution to obtaining a correct positioning of the numbers in the cells of a puzzle, with proper previous filtering of the combination of the numbers in the sets of sums. The pre-computation phase has given great improvements to the SAHS algorithm in order to reduce the search space. The proposed solution has been experimentally evaluated on various grids with different size and the results shown that the algorithm is very robust and always converged to the solution.

Keywords

Puzzle; grid-based games; metaheuristic; Harmony Search Algorithm

Introduction

Games are an integral part of the human's social life since the ancient times. Among all types of games, logic puzzles and board and games, historically have been considered very popular and were played not only to pass the time, but also to signify the wealth and status of the players. From scientific perspective logical puzzels have been analysed and studied from many matematiciants all over the world. Recently, with the technological development they have also become very attractive in the fields of computer science and artificial intelligence, since they have contributed to evaluation of performances of various metaheuristic algorithms and development of new ones.

One particular subset of logical puzzles are those played in a grid-based environment. Examples of grid-based puzzles include: Sudoku and its variants (Arnold et al. 2012), Slitherlink (Liu, 2012), Peg-Solitaire (Ravikumar et al., 2004), Blocks (Slaney and Thiebaux, 2001), Mastermind (Stuckman and Zhang, 2006). Many methodologies and algorithms for solving various grid-based puzzles have also been proposed (Ortiz-García et al. 2008), (Batenburg and Kosters, 2004), (Wiggers, 2004), (Salcedo-Sanz et al. 2007).

Solving this kind of puzzles, besides the logic, often includes various constraints implied by the game grid, to which the solution should be fitted. It has been proven that many grid-based puzzles are NP-complete problems i.e. they require exponential time to be solved optimally. On the other hand metaheuristic algorithms are considered an important alternative to solve this kind of problems.

In this work we are presenting a Self Adapting Harmony Search (SAHS) metahauristic based solution to one world popular grid-based Japaneese game know as Kakuro. To the best of authors' knowledge, none research exists up to date, describing the solution of the Kakuro grid-based puzzle using Harmony Search based metaheuristic algorithm. The proposed solution has been experimentally evaluated on various grids with different size and the results are presented.

Kakuro – Game Description

Kakuro is a type of grid-based logic puzzle that is usually known as a mathematical transliteration of the crossword. It is composed of a simple grid which dimensions can vary from 3x3 matrix through to a grid with large dimensions NxM ($N,M >= 3$). The grid is composed of "blank" and "occupied" cells, white and black correspondently. By rule the top row and leftmost column are occupied and are not used for the gameplay while the rest od the grid contains entry series of white cells devided by some black cells. Some of the black cells contain a diagonal slash from upper-left to lower-right corner and a number in one or both halves, denominated as clues (Fig. 1).

The main aim of the game is to fill in all the white cells with numbers from 1 to 9. All cells in the same row and all cells in the same column have to contain different number. The sum of the numbers of a row or a column has to match a predefined clue, associated with it.

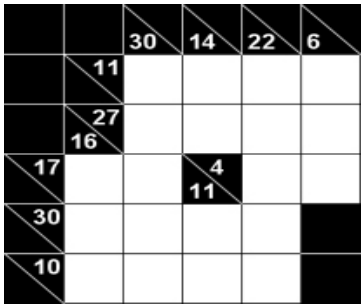


FIG. 1 KAKURO PUZZLE GRID.

The puzzle is solved when all the white cells are filled in according to the game rules and respecting the given constraints. The solution of the Kakuro puzzle given in Fig. 1 is presented in Fig. 2. The solution of the game mainly depends on its structure, and usually more blank cells the grid contains, the harder it is to be solved.

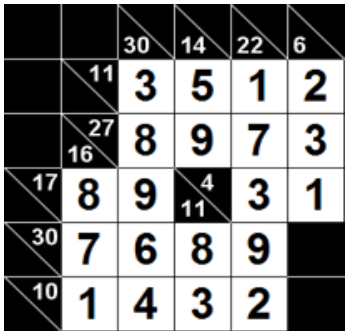


FIG. 2 SOLVED KAKURO PUZZLE.

Various solving techniques of Kakuro puzzles exists and some of them include: Lone square, Cross reference, Combo reference, Filled areas, Eliminating duplicates. There are two kinds of mathematical symmetry readily identifiable in kakuro puzzles: minimum and maximum constraints are duals, as are missing and required values.

Related Work

There have been several previous attempts for solving the Kakuro puzzle in the past few years. The NP-completeness of the puzzle has been proved using various techniques (Seta, 2002), and some of the latest proofs include using SAT solvers (Ruepp and Markus, 2010). Some researches (Davis et al., 2009) provided methods using heuristic approaches to improve a backtracking solving technique and introduced a pruning approach to decrease the time of obtaining a solution. They have also emphasized the utilization of these techniques in coding theory and applying them in real-world applications, which proved the importance of developing such algorithms.

A Kakuro puzzle can be easily observed as a constraint satisfaction problem (Simonis, 2008). It has been proven that by using generalized arc consistent (GAC) version of all-different sums constraint reduces the necessity of searching, compared to MIP (Achterberg et al., 2006) and SAT techniques (Eén and Sörensson, 2004). For the first time it has been shown that a Nested Monte-Carlo search of level 2 can give satisfying results for the purposes of solving such puzzles (Cazenave, 2010).

Harmony Search (HS) is an optimization technique which is analogous to the process of improvisation of Jazz musicians (Yang, 2009) and has been widely used in the last decade for optimization purposes. HS has already been used for solving Sudoku puzzles (Geem, 2007) and has proved to provide good solutions in 285 iterations in less than 9 seconds. This is a motivation to apply HS to other challenging NP-complete puzzles, namely Kakuro, which is described in the following.

Harmony Search (HS) Algorithm Description

In the Harmony Search (HS) algorithm, an optimal solution is constructed by using all available combinations present in memory, i.e. the Harmony Memory (HM), and then uses randomization techniques to select candidate values to obtain new solutions towards the optimal value of a fitness function. HS constructs the best possible solution in several steps:

1) *Harmony Memory (HM) initialization.* The matrix of all possible solutions has to be initialized to randomly generated values, all specified in a predefined range of numbers. If n presents the dimension of the candidate vectors, whereas *N* presents the size of the memory, then *HM* could be presented as follows:

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^N & x_2^N & \cdots & x_n^N \end{bmatrix} \quad (1)$$

2) *Candidate solution improvisation.* This step of the process is dependent of several parameters. The first is Harmony Memory Consideration

Rate (HMCR), which indicates the probability the next constructed solution to be selected from HM, whereas (1-HMCR) would indicate the probability this new solution to be generated randomly from the range of possible values. The next crucial parameter, if one needs a certain variability of the chosen solutions and ability to influence the memory values, is the Pitch Adjustment Rate (PAR). Thus, PAR indicates the probability of a candidate solution being subject to change. Furthermore, the maximum permitted variability in pitch adjustment is given by the bandwidth (BW) parameter. Finally, the new generated solution would have the form $\left[x_1^{'} \quad x_2^{'} \quad ... \quad x_n^{'} \right]$, where n is the dimension of the candidate vectors.

3) *Comparison*. The new constructed solution is then compared to all the vectors present in the HM. If this new vector provides better fitness function evaluation then the worst present solution in HM, these two solutions are exchanged, so the new solution is temporarily inserted in memory, and the worst is discarded. If the value of the fitness function does not satisfy the previously stated condition, the new candidate solution is discarded. If the termination criteria is not met, such as number of iterations or a concrete value of the fitness function, steps 2) and 3) are repeated. Otherwise, the algorithm terminates.

Applying Harmony Search To Solving Kakuro Puzzles

In this paper, we introduce a novel approach to solving Kakuro puzzles. For the purposes of this research, before the Harmony Search (HS) stage of the algorithm, several eliminations of all possible combinations need to take place. This is crucial to the time of execution of the algorithm, as well as providing the best solution with as fewer iterations as needed. The proposed algorithm includes the several different stages:

1) Initialization of combinations

At first, all of the present sums in the puzzle are examined, both horizontally and vertically. For each of the sums a combination of the elements that can form the concrete sum is constructed and is saved as a set of numbers for the next stages of computation. For instance, if the sum that is of our current interest is the number 11 (eleven), it is known that it can be obtained by using the pairs of sums (2,9), (3,8), (4,7) and (5,6), so the combination set for the number 11 is {2,3,4,5,6,7,8,9}.

Computing intersections of combination sets

For all of the free cells in the puzzle, intersections of the combination sets of sums that intersect the cells are computed. This is imperative in order to reduce the combinations needed to construct a working set for the Harmony Search (HS) stage of the algorithm. For example, in Fig. 3 one can see that the red cell is affected by the combination of sums for the numbers 11 (vertically) and 8 (horizontally). For the number 11 we have the combination set of {2,3,4,5,6,7,8,9}, and for the number 8 we have the combination set of {1,2,3,5,6,7}. Therefore, the intersection of the two sets will be {2,3,5,6,7}, which now contains possible values for the marked cell.

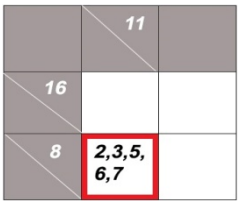


FIG. 3 INTERSECTION OF SUMS FOR THE RED CELL. THIS CELL IS AFFECTED BY THE COMBINATIONS FOR THE NUMBERS 11 AND 8.

Removing numbers from sets of possible values by identifying unique numbers

For all of the free cells in the puzzle, a method for eliminating elements of the sets of candidate numbers by recognizing unique values is implemented. For instance, if a concrete cell is observed that has a value that is not present in the sets of possible values for all of the other cells horizontally or vertically, then it is clear that all of the other values in the cell can be removed, thus leaving only the concrete number a solution to the observed cell. Such an example is shown is Fig. 4, where the fourth cell has one unique number (the number 2), that is not present in the sets of the possible values for the other cells horizontally, consequently making this number a solution to that cell.

FIG. 4 REMOVING ELEMENTS FROM CELLS WITH UNIQUE NUMBERS. THE NUMBER 2 IS UNIQUE FOR THE LAST CELL.

Self-Adaptive Harmony Search

Finally, the sets of possible values are constructed and

the HS stage of the algorithm is initialized. For the value of the HMCR parameter a value of 0.9 is chosen as a typical value belonging to the recommended range of values in [0.75, 0.99]. For this concrete stage, a self-adaptive HS algorithm (SAHS) is utilized, which reduces the necessity of fine tuning the parameters manually (Alia and Rajeswari, 2011). Thus, the PAR and BW parameters are changed dynamically through the iterating part of the SAHS algorithm. The PAR parameter is linearly increased as shown in the Eq. 2:

$$PAR(gn) = PAR_{\min} + \frac{(PAR_{\max} - PAR_{\min})}{NI} \times gn \qquad (2)$$

where *gn* indicates the number of the current generation (iteration), *PAR(gn)* the pitch adjustment for the gn iteration, PAR_{min} and PAR_{max} the value range of the pitch adjustment, and *NI* the maximum number of iterations (Alia and Rajeswari, 2011). For this paper, the following values are set: $PAR_{min}=0.05$ and $PAR_{max}=0.5$. The BW parameter is exponentially decreased over the number of iterations, which is indicated by the Eq. 3:

$$BW(gn) = BW_{\min} + \frac{BW_{\max} - BW_{\min}}{NI} \times gn \qquad (3)$$

where *gn* indicates the number of the current generation (iteration), *BW(gn)* the bandwidth for the gn iteration, BW_{min} and BW_{max} the value range of the bandwidth, and *NI* the maximum number of iterations (Alia and Rajeswari, 2011). For this paper, the following values are set: $BW_{min}=0.01$ and $BW_{max}=0.3$.

The candidate solutions obtained from HM ready to be evaluated by the fitness function follow the next equation (Eq.4)(Geem, 2010):

$$x_i' \leftarrow \begin{cases} x_i' \in \left[x_i^1 \quad x_i^2 \quad \cdots \quad x_i^{HMS} \right] & w.p. \quad HMCR \\ x_i' \in X_i & w.p. \quad (1-HMCR) \end{cases} \qquad (4)$$

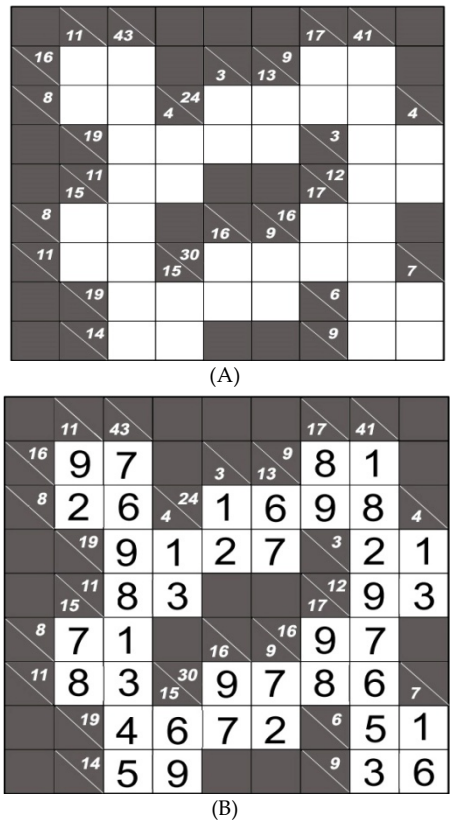
where x_i' is the new candidate vector, X_i is the set of all integer values in range [*0; maximumCombinationNumbers-1*], indicating the index of the candidate number in the combination set for the observed free cells, whereas *maximumCombinationNumbers* is the size of the combination set. Eq. 4 presents that with probability of HMCR a new solution will be constructed from the HM, whereas with probability of (1-HMCR) this solution will be constructed randomly from the value range for the possible values.

For the main purposes of the SAHS algorithm, the fitness function for evaluating the candidate vectors is defined as follows:

$$f(.) = \sum_{v \in V} \left| TV(v) - CV(v) \right| + \sum_{h \in H} \left| TH(h) - CH(h) \right| \qquad (5)$$

For the first sum of the fitness function, *V* is the set of vertical sums in the puzzle, v is a vector from the set *V*, *TV(v)* is the expected vertical sum for the vector *v*, whereas *CV(v)* is the computed vertical sum obtained from the candidate values in the new solution for that vector. Analogously for the second sum, *H* is the set of horizontal sums in the puzzle, *h* is a vector from the set *H*, *TH(h)* is the expected horizontal sum for the vector *h*, whereas *CH(h)* is the computed horizontal sum obtained from the candidate values in the new solution for that vector.

For our research purposes, several puzzles with sizes of 5x5, 6x6, 8x8, 10x10, 12x12 and 15x15 have been examined. The algorithm has been implemented in Java programming language and the test cases have been examined on a computer machine with Intel® Core i5 processor having frequency of 2.53 GHz, 4GB of RAM, and 64bit Windows 7 operating system. An example puzzle solved by using this algorithm is shown in Fig. 5.



(A)

(B)

FIG. 5 (A) EXAMPLE PUZZLE OF SIZE 8X8. (B) SOLUTION TO THE PUZZLE IN (A).

The results of applying the SAHS to the test cases expressed in milliseconds and number of iterations are shown in Table 1 and Table 2, respectively.

TABLE 1 RESULTS OF APPLYING SAHS TO KAKURO PUZZLES IN MILLISECONDS

Grid Sizes	Test cases				
	Case1	Case2	Case3	Case4	Case5
5x5	27	21	22	23	31
6x6	30	33	34	32	38
8x8	57	48	70	52	57
10x10	99	88	96	101	86
12x12	163	154	482	159	133
15x15	940	1407	2497	239	256

TABLE 2 RESULTS OF APPLYING SAHS TO KAKURO PUZZLES IN NUMBER OF ITERATIONS.

Grid Sizes	Test cases				
	Case1	Case2	Case3	Case4	Case5
5x5	100	100	100	100	100
6x6	100	100	100	100	100
8x8	100	100	100	100	100
10x10	100	100	100	100	100
12x12	100	100	8000	100	100
15x15	10000	20000	50000	100	100

These results have proved that SAHS can be provide a great alternative to solving Kakuro puzzles and an optimal solution can be always found based on the number of iterations previously defined by the algorithm. The time execution of the algorithm proved to be acceptable when it comes to computing an optimal solution, which makes it applicable to real-world applications.

Conclusions

Metaheuristic approach to solving a Kakuro puzzle has been introduced. It has been shown that the Self-Adapting Harmony Search (SAHS) technique can provide a time-efficient solution to obtaining a correct positioning of the numbers in the cells of a puzzle, with proper previous filtering of the combination of the numbers in the sets of sums. The pre-computation phase has given great improvements to the SAHS algorithm in order to reduce the search space. This makes the algorithm applicable to future real-world applications.

REFERENCES

Achterberg, Tobias, Thorsten Koch, and Alexander Martin. "MIPLIB 2003."Operations Research Letters 34.4 (2006): 361-372.

Alia, Osama Moh'D., and Rajeswari Mandava. "The variants of the harmony search algorithm: an overview." Artificial Intelligence Review 36.1 (2011): 49-68.

Arnold, E., R. Field, J. Lorch, S. Lucas, and L. Taalman. "Nest graphs and minimal complete symmetry groups for magic Sudoku variants." (2012).

Batenburg B, Kosters W (2004) A discrete tomography approach to Japanese puzzles. In: Proceedings of the Belgian-Dutch conference on artificial intelligence, pp 243–250

Cazenave, Tristan. "Monte-Carlo Kakuro." Advances in Computer Games(2010): 45-54.

Davies, R. P., P. A. Roach, and S. Perkins. "Automation of the Solution of Kakuro Puzzles." Research and Development in Intelligent Systems XXV(2009): 219-232.

Eén, Niklas, and Niklas Sörensson. "An extensible SAT-solver." Theory and Applications of Satisfiability Testing. Springer Berlin/Heidelberg, 2004.

Geem, Zong Woo. Recent advances in harmony search algorithm. Vol. 270. Springer, 2010.

Geem, Zong. "Harmony search algorithm for solving sudoku." Knowledge-Based Intelligent Information and Engineering Systems. Springer Berlin/Heidelberg, 2007.

Liu, Tung-Ying, I. Wu, and Der-Johng Sun. "Solving the Slitherlink Problem." In Technologies and Applications of Artificial Intelligence (TAAI), 2012 Conference on, pp. 284-289. IEEE, 2012.

Ortiz-García E, Salcedo-Sanz S, Pérez-Bellido AM, Portilla-Figueras A, Yao X (2008). Solving very difficult Japanese puzzles with a hybrid evolutionary-logic algorithm. Lect Notes Comput Sci 5361:360–369

Ravikumar, B. (2004). Peg-Solitaire, String Rewriting Systems and Finite Automata. Theoretical Computer Science, Vol. 321, Nos. 2–3, pp. 383–394.

Ruepp, Oliver, and Markus Holzer. "The computational complexity of the KAKURO puzzle, revisited." Fun with Algorithms. Springer Berlin/Heidelberg, 2010.

Salcedo-Sanz S, Portilla-Figueras J, Pérez.Bellido A, Ortiz-García E, Yao X (2007) Teaching advanced features of evolutionary algorithms using Japanese puzzles. IEEE Trans Edu 50(2):151–155

Seta, T. A. K. A. H. I. R. O. "The complexity of CROSS SUM." IPSJ SIG Notes, AL-84 (2002): 51-58.

Simonis, Helmut. "Kakuro as a constraint problem." Proc. seventh Int. Works. on Constraint Modelling and Reformulation (2008).

Slaney, J. and Thiebaux, S. (2001). Blocks World Revisited.

Artificial Intelligence, Vol. 125, pp. 119–153.

Stuckman, J. and Zhang, G.-Q. (2006). Mastermind is NP-Complete. INFOCOMP Journal of Computer Science, Vol. 5, pp. 25–28.

Wiggers W (2004) A comparison of a genetic algorithm and a depth first search algorithm applied to Japanese nonograms. In: Proceedings of the 1st Twente Student conference on IT, pp 1–6

X.-S. Yang, "Harmony Search as a Metaheuristic Algorithm", in: Music-Inspired Harmony Search Algorithm: Theory and Applications (Editor Z. W. Geem), Studies in Computational Intelligence, Springer Berlin, vol. 191, pp. 1-14 (2009).

Stojanche Panov obtained his Bachelor's degree in Computer Science in 2011 at the Faculty of Computer Science, "Goce Delchev" University – Stip. Currently, he's a postgraduate student at the same faculty at the robotics and intelligent systems division.

He's a former Microsoft Student Partner and has performed several public presentations on Microsoft technologies. His main research interests are robotics, artificial intelligence algorithms, cryptography and game theory.

Saso Koceski obtained his PhD in robotics and artificial intelligence in 2009 from the University of L'Aquila, Italy. Currently he is an assistant professor at the Faculty of Computer Science, University "Goce Delcev"-Stip, Macedonia and head of the Institute of Computer Science at University "Goce Delcev"-Stip.

He is an author or co-author of more than 50 refereed journal and conference papers and book chapters. He serves as an editor and reviewer in several SCI journals. He has been involved in several international and national scientific and applicative projects (including EU-TEMPUS and EU-COST) as a project coordinator or participant. His current research interests are focused in the field of robotics and artificial intelligence, bioinformatics, HCI and medical imaging.