Mathematica
Balkanica

New Series Vol. 25, 2011, Fasc. 5

Application of Exact and Heuristic Methods to Magic Square Problem

Aleksandar Đenić

The Constraint Satisfaction Problem is a mathematical problem defined as a set of objects whose state needs to satisfy a number of constraints. The *CSP* is defined as a set of variables $X_1, X_2, \ldots, X_n$, and a set of constraints $C_1, C_2, \ldots, C_m$. The solution of the problem is a set of mappings of values from given domains to variables which satisfies all the constraints. In this paper, the magic square problem is solved with exact and heuristic solvers and results are compared and analyzed. Exact techniques are represented through creating a model for Microsoft Solver Foundation and IBM ILOG CPLEX solver, and a heuristic by applying genetic algorithm and variable neighborhood search.

MSC2010: 90C90

Key Words: Constraint Satisfaction Problem, Magic Square, Microsoft Solver Foundation, Ibm Ilog Cplex, Genetic algorithm, Variable Neighborhood Search

1. Introduction

The Constraint Satisfaction Problem (CSP) $\mathcal{P}$ is a triple $\mathcal{P} = \langle X, D, C\rangle$ where X is an n-tuple of variables $X = \langle x_1, x_2, \ldots, x_n\rangle$, D is a corresponding n-tuple of domains $D = \langle D_1, D_2, \ldots, D_n\rangle$ such that $x_i \in D_i$, C is a m-tuple of constraints $C = \langle C_1, C_2, \ldots, C_m\rangle$. A constraint C_j is a pair $\langle R_{S_j}, S_j\rangle$ where R_{S_j} is a relation on the variables in $S_j = scope(C_i)$. In other words, R_i is a subset of the Cartesian product of the domains of the variables in S_i.

A solution to the CSP $\mathcal{P}$ is an n-tuple $A = \langle a_1, a_2, \ldots, a_n\rangle$ where $a_i \in D_i$ and each C_j is satisfied in that R_{S_j} holds on the projection of A onto the scope S_j. In a given task it may be required to find the set of all solutions, $sol(\mathcal{P})$, to determine if that set is non-empty or just to find any solution, if one exists. If the set of solutions is empty the CSP is unsatisfiable. More details about CSP are given in [8].

Algorithms for solving problems of mathematical optimization can be complete and incomplete. Complete or exact algorithms find an optimal solution, together with the proof of its optimality, or they give proof that the required solution does not exist. Incomplete, or heuristic algorithms are used for finding the approximation of the optimal solution. These algorithms can not prove that the problem does not have a solution nor that the found solution is optimal. However, incomplete algorithms are often more efficient in finding the solution if it exists. In this paper, the magic square problem is solved with exact and heuristic methods and results are compared and analyzed.

The magic square puzzle consists in placing on a $N \times N$ square all the numbers in $\{1, 2, \ldots, N^2\}$ such as the sum of the numbers in all rows, columns and the two diagonal are the same. It can therefore be modeled in CSP by considering N^2 variables with initial domains $\{1, 2, \ldots, N^2\}$ together with linear equation constraints and a global all different constraint stating that all variables should have a different value. The constant value that should be the sum of all rows, columns and the two diagonals can be easily computed to be $N \cdot (N^2+1)/2$. Earlier, the magic square problem is solved in [1].

2. Implementation and results

2.1. Microsoft solver foundation

The Microsoft Solver Foundation (MSF) is a library of functions for mathematical programming, modeling and optimization, developed by Microsoft DevLabs.

A problem is defined by a model, by assigning variables and their domains, constraints, goals and data being processed. The solution is obtained by solving the model with one of the solvers: simplex solver ([2]), solver which uses interior point method ([5], [6]) and CSP solver.

In this paper, the CSP solver is used. It uses backtracking techniques ([4]), local search and metaheuristics ([3]).

More detailed description of MSF is given in [9].

The Magic square model of order n defined for MSF CSP solver has n^2 variables $x_{11}, x_{12}, \ldots, x_{nn}$, where every variable represents one value from the square. Each variable takes value from the domain:

$$x_{11}, x_{12}, \ldots, x_{nn} \in \{1, 2, \ldots, n^2\}.$$

The first constraint that is being introduced is that all variables have to be different:

$$Unequal\,(x_{11}, x_{12}, \ldots, x_{nn}).$$

The variable b, which represents the value of the required sum of each row, column and diagonal is given with:

$$b = \frac{n \cdot (n^2+1)}{2}.$$

Other constraints are:

- Sum of every row is b

$$\sum_{j=1}^{n} x_{ij} = b, \qquad i \in \{1, 2, \ldots, n\},$$

- sum of every column is b

$$\sum_{j=1}^{n} x_{ji} = b, \qquad i \in \{1, 2, \ldots, n\},$$

- sum of every diagonal is b

$$\sum_{i=1}^{n} x_{ii} = b, \qquad \sum_{i=1}^{n} x_{i(n-i+1)} = b\,.$$

While solving this model, the first time the solver finds a solution which satisfies all the constraints it stops with executing the algorithms and returns that solution.

2.2. CPLEX

IBM ILOG CPLEX (CPLEX) is an optimization software package for solving problems of mathematical programming. CPLEX solves problems of linear and quadratic programming. It uses the simplex method ([2]) and interior point methods ([5], [6]).

In this model, there is no constraint which describes the uniqueness of the variables, so it is introduced through a series of constraints:

$$|x_{ij} - x_{lk}| > 0, \qquad (i, j) \neq (l, k)\,.$$

This way of assigning constraints is not efficient because for a magic square of order n, $\frac{n^2 \cdot (n^2-1)}{2}$ different constraints is given. Also, the variable b represents the value of the required sum of every row, column and diagonal:

$$b = \frac{n \cdot (n^2 + 1)}{2}\,.$$

The model requires a goal function which needs to be minimized or maximized. For that reason, a term of error is introduced. The deviation of the sum of every row, column and diagonal from the required values adds up to the error function. In the moment when all the deviations are equal to zero, magic square is found. Error functions on rows, columns and diagonals are given with additional variables e_1, e_2 and e_3, respectively:

- total error on rows

$$e_1 = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} x_{ij} - b \right| ,$$

- total error on columns

$$e_2 = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} x_{ji} - b \right| ,$$

- diagonal error

$$e_3 = \left| \sum_{i=1}^{n} x_{ii} - b \right| + \left| \sum_{i=1}^{n} x_{i(n-i+1)} - b \right| .$$

The solver minimizes the error function:

$$f(x_{11}, x_{12}, \dots, x_{nn}) = e_1 + e_2 + e_3 .$$

Because of the large number of constraints required to describe variables uniqueness, linear solvers are not suitable for given type of problems.

2.3. Genetic algorithm

For solving the problem with genetic algorithm ([3]), the open source library GA Framework[1] is used. It supports the multiprocessor execution of the algorithm. In its implementation, the following steps are iteratively executed:

1. sorting of the individuals according to the fitness function;
2. selection of the individuals for reproduction;
3. crossover;
4. replacing of not fit individuals with new ones;
5. mutation.

In this paper, a population of 150 units is used. Initial population is randomly generated. A normalized objective function on the $[0, 1]$ interval is used as a fitness function (the best individual has fitness 1, the worst one has fitness 0). If the number of identical individuals is greater than the number of allowed identical individuals, the fitness value for excess individuals gets set to

[1] `http://code.google.com/p/gaframework/`

0. Also, if the number of individuals with same value of fitness the function gets greater than the maximum number of individuals with same fitness value, excess individuals will get their fitness function values set to 0. The selection used is a finely graded tournament selection. 50 individuals are chosen for parents, which, using the operator of crossover, give 50 new individuals.

The search space is the set S of all permutations of numbers $\{1, 2, \ldots, n^2\}$. Every point from the set S uniquely identifies a matrix $n \times n$ which you get by writing the numbers in rows of the matrix, respectively. The objective function is defined as:

$$.f : S \to \mathbb{Z},$$

and has the value of the sum of the deviation of the sum of every column, row and diagonal from the searched value in the magic square.

The goal is to minimize the objective function, i.e. to find the permutation for which the deviation is equal to zero. That permutation is the searched magic square of order n.

Every individual in the population codes one permutation of set S. Chromosome consists of a series of numbers of total length n^2 which represents the searched permutation.

The crossover process is defined as follows:

- With probability of 0.6 one parent will give one offspring, another will give another offspring (without real crossover) by replacing 2 randomly chosen numbers.
- With the probability of 0.2 both children will have, in places where their parents overlap, same values, and other values will be randomly chosen in a way which preserves permutation.
- With the probability of 0.2 daughter will get one part of mother's chromosome and one part of father's chromosome, and the son will get other parts of those chromosomes. In that case, it is possible to get an invalid permutation and those individuals need to be fixed.

The mutation is defined as a permutation of 2 numbers in the chromosome.

Parameters used are:

- The maximum number of identical individuals: 2,
- The maximum number of individuals with the same value of objective function: 20,
- The expected average tournament size: 5.2,
- The probability of crossover: 0.8,
- The probability of mutation: $0.5/n^2$,

- The frozen bit factor: 4.

2.4. Variable neighborhood search

For solving the magic square problem with VNS technique ([3], [7]), a specific VNS algorithm which corresponds to the problem was implemented.

For the purposes of this paper, the VNS technique applied had a stopping criteria of one thousand executed iterations. The initial solution for the reduced VNS was randomly selected, and the initial solution for basic VNS is a solution of the reduced VNS. Same collection of neighbourhood structures was used in reduced VNS, local search and basic VNS algorithms.

The search space and objective function, S and f, respectively, which are used in the variable neighborhood search method are identical to those used in the genetic algorithm. The goal is to minimize the objective function, i.e. to find the permutation for which the error equals zero. That permutation is the magic square of order n.

For first neighborhood of a given point x from the solution space, a set of points which differ from x in exactly two numbers is used. By replacing two numbers in the given permutation, from the point x we get the point x', and $x' \in N_1(x)$ stands.

After replacing 2 numbers in a permutation, the value of objective function needs to be recalculated. That process can be optimized by using the fact that the new point is in the neighborhood of the old point. Every point contains information about the error of every row, column and diagonal, and the total error. The values in the sequence of errors of rows gr, columns gk and diagonals gd can be positive and negative, depending on whether the sum is greater than or less than the desired sum. The total error is then equal to the sum of absolute values of all errors in columns, rows and diagonals.

If the point x_{ij} is replaced by x_{lk} then:

- From the total error, the error of order i is substracted: $g = g - |gr_i|$,
- The new error of order i is calculated: $gr_i = gr_i - x_{ij} + x_{lk}$,
- The error of order i is added to the total error $g = g + |gr_i|$,
- From the total error, the error of order l is substracted: $g = g - |gr_l|$,
- The new error of order i is calculated: $gr_l = gr_i + x_{ij} - x_{lk}$,
- The error of order l is added to the total error $g = g + |gr_l|$.

In a similar way, the errors for columns and diagonals are modified. In that way, the objective function is calculated with time complexity of $\mathcal{O}(1)$, which greatly improves the total calculation speed.

In the shake algorithm, k consecutive switches of pairs of points in permutation are conducted. A number which was switched once, cannot be switched again. A switch is defined as follows:

- With the probability of 0.1 the number in the permutation is selected randomly, then the best one to switch with it is found
- With the probability of 0.9 both numbers are randomly chosen.

2.5. Results

All four mentioned techniques are tested for solving the magic square problem of order 3 to 21. All tests are conducted on a computer with Intel Core i7 860 and every instance had execution time of up to one hour. CPLEX solver solved two smallest instances, MSF CSP solved first five instances, the genetic algorithm solved fourteen, and the variable neighborhood search solved nineteen instances.

Table 1 gives all the results.

Dimension	VNS[s]	GA[s]	MSF[s]	CPLEX[s]
3	0.02	0.01	0.10	0.03
4	0.40	0.05	0.10	4.28
5	0.25	1.32	0.23	—
6	0.34	3.27	3271.94	—
7	3.00	15.09	0.14	—
8	7.28	4.17	—	—
9	1.13	117.23	—	—
10	12.44	31.00	—	—
11	13.23	16.55	—	—
12	49.13	72.39	—	—
13	14.56	1027.54	—	—
14	9.16	95.52	—	—
15	156.84	554.87	—	—
16	154.56	1405.69	—	—
17	302.16	—	—	—
18	1067.04	—	—	—
19	1596.38	—	—	—
20	870.74	—	—	—
21	2577.42	—	—	—

Table 1: Magic Square – results.

3. Conclusion

In this paper, the magic square problem, which belongs to the class of constraint satisfaction problems is solved with exact and heuristic methods. The exact solvers used are CPLEX and Microsoft Solver Foundation, and the heuristic methods applied are genetic algorithm and VNS.

As assumed, heuristics gave better results. Magic squares of order higher than three exist and in problems which have solutions heuristics, in general, faster find those solutions.

References

[1] P. Codognet, D. Diaz. Yet Another Local Search Method for Constraint Solving, Stochastic Algorithms: Foundations and Applications, **2264**, 2001, 342-344.

[2] G. B. Dantzig. Linear Programming and Extensions, Princeton University Press, New Jersey, 1963.

[3] M. Gendreau, J.-Y. Potvin. Handbook of Metaheuristics, Second Edition, Springer, New York, 2010.

[4] S. Golomb, L. Baumert. Backtrack programming, JACM, **12**(3), 1965, 516-524.

[5] N. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming, Combinatorica, **4**, 1984, 373-395.

[6] S. Mehrotra. On the Implementation of a Primal-Dual Interior Point Method, SIAM J. Optimization, **2**, 1992, 575-601.

[7] N. Mladenovic, P. Hansen. Variable neighborhood search, Computer & Ops Res, **24**, 1997, 1097-1100.

[8] F. Rossi, P. v. Beek, T. Walsh. Handbook of Constraint Programming, Foundations of Artificial Intelligence, Amsterdam, 2006.

[9] L. Surhone, M. Timpledon, S. Marseken. Solver Foundation, VDM Verlag Dr. Mueller AG & Co. Kg, 2010.

Aleksandar Đenić
Faculty of Mathematics
University of Belgrade
Studentski trg 16,
11000 Belgrade, SERBIA
E-mail: djenic@matf.bg.ac.rs