


Start-up and the results of the volunteer computing project RakeSearch

Maxim Manzyuk¹, Natalia Nikitina², and Eduard Vatutin³

¹ Internet portal BOINC.ru, Moscow, Russia,
hoarfrost@rambler.ru

² Institute of Applied Mathematical Research, Karelian Research Center of the
Russian Academy of Sciences, Petrozavodsk, Russia,
nikitina@krc.karelia.ru

³ Southwest State University, Kursk, Russia,
evatutin@rambler.ru

Abstract. In this paper we describe the experience of setting up a computational infrastructure based on BOINC middleware and running a volunteer computing project on its basis. The project is aimed at characterizing the space of diagonal Latin squares of order 9 in the form of an ensemble of orthogonality graphs, previously not addressed. We implement the search for row-permutational squares orthogonal to an initial one, which allows to reconstruct the full graphs. We provide the developed application to search for orthogonal pairs of the squares and describe the obtained results. The results prove the efficiency of volunteer computing in unveiling the structure of the space of diagonal Latin squares.

Keywords: Desktop Grid · Volunteer computing · BOINC · Orthogonal diagonal Latin squares · Orthogonality graph

1 Introduction

1.1 Orthogonal Diagonal Latin Squares

A Latin square (further LS) of order N is an $N \times N$ array filled with elements from a finite set of size N in such a way that all elements within a single row or single column are distinct. A Latin square is called diagonal (further DLS) if all elements in both its main diagonal and main back-diagonal are distinct.

Two diagonal Latin squares $A = \{a_{ij}\}$ and $B = \{b_{ij}\}$ are called orthogonal mates (further ODLSs) or Graeco-Latin squares if all ordered pairs (a_{ij}, b_{ij}) , $1 \leq i, j \leq N$, are distinct. A DLS is called normalized if the elements of its first row are sorted in ascending order (see Figure 1). It is easy to show that by means of a bijective mapping of the elements one can normalize any correct DLS, and the corresponding set of DLSs forms an equivalence class of $N!$ of them. For a number of problems, the squares within the equivalence class do not differ, since they all possess the same properties (existence/absence of an orthogonal mate,

1	3	2	0	4
4	2	0	1	3
0	4	3	2	1
3	0	1	4	2
2	1	4	3	0

→

0	1	2	3	4
4	2	3	0	1
3	4	1	2	0
1	3	0	4	2
2	0	4	1	3

Fig. 1. Normalization of a DLS

the number of transversals etc.), which allows to significantly reduce the runtime in any corresponding computational experiment.

Latin squares find their applications in different areas [1]: in graph theory [2], the design of experiments [3], tournament scheduling [4], cryptography [5], encoding methods with error control for information transmission [6]. In theoretical combinatorics, a known unsolved mathematical problem is to find a triple of mutually orthogonal DLSs of order 10 or to prove that it does not exist [7].

In applications, it is important to know the structures of ODLs subsets. However, the spaces of DLSs of order 9 or more have not been explicitly enumerated or described due to their large size. The presented work is aimed at directed research of the space of DLSs of order 9 using volunteer computing.

The systems of DLSs of order 9 connected by orthogonality relation have not been described before, excepting the case when all squares of a set are mutually orthogonal. Construction of ODLs of orders 7 and 8 showed that row-permutational ODLs can constitute parts of such systems. In the proposed work, the search is performed among the subset of such squares that can be obtained by rows permutations (with fixed first row) and are orthogonal to the initial square.

1.2 Desktop Grids

Desktop Grid is a distributed computing system that gathers together desktop computers, servers and/or other heterogeneous computational resources connected by the Internet or a local network and working for the Desktop Grid in their idle time. Traditional high-performance computing systems can be integrated into Desktop Grids as well [8]. The BOINC middleware [9] is often considered a de-facto standard for organizing Desktop Grids, as it is being actively developed and has been successfully used in many computational projects since 1997.

The BOINC system has a server-client architecture: a server distributes independent tasks to the nodes which perform computations and return the results to the server for further processing. Such division of a large resource-demanding computational problem into many independent tasks allows to solve it efficiently in a shorter time.

1.3 Search for Orthogonal Diagonal Latin Squares Using Desktop Grids

There are plenty of works that have used computers for finding or enumerating combinatorial structures based on Latin squares [10–14], in particular BOINC-based Desktop Grids [15–18].

The problem of characterizing subsets of ODLs (as well as DLSs and LSs) fits well for implementation in Desktop Grids. The volunteer computing project SAT@home [19] searches for the pairs of ODLs of order 10 and their systems using SAT methods within solving the problems of cryptanalysis. The volunteer computing project Gerasim@home [20] performs search for the systems of ODLs of order 10 and estimate some characteristics of DLSs of small order such as the minimal and maximal numbers of their transversals, number of symmetric and double symmetric squares, etc.

At the same time, the systems of ODLs of order 9 have not been described, excepting the case when all squares of a set are mutually orthogonal. In this work, we describe the volunteer computing project RakeSearch [21] aimed at finding and describing such systems. The project is implemented basing on BOINC middleware within the grid segment of the Center for Collective Use of Karelian Research Center of the Russian Academy of Sciences [22]. The experience of building the computational infrastructure and porting the developed search application to it are described in [23].

In the rest of the paper we describe the details of implementing the volunteer computing project and the obtained results.

2 Setup of the BOINC-based Desktop Grid

2.1 Server Infrastructure

At the moment, the project server works in a virtual machine which has 40 Gb hard drive and 8 Gb RAM. It is more than sufficient for supporting its work. On the average, 120 thousand of workunits are being processed every day; the average real performance of the project fluctuates within the range from 125 to 175 TeraFLOPS.

2.2 An Approach to Fast Discovery of Orthogonal Diagonal Mates

The total number of Latin squares of order 9 is 377 597 570 964 258 816 [24]. A classical method to search for ODLs is the Euler-Parker approach which results in relatively slow discovery rate (about 2000 DLSs per second for order 9). It was noticed that the search among row-permutational squares (the ones obtained from an initial square by permutations of its rows) allows to find ODLs much faster, resulting in the rate about 44–92 thousand DLSs per second depending on the optimization level. Such approach became the basis for the RakeSearch project. Figure 2 shows an example of row-permutational ODLs of order 9.

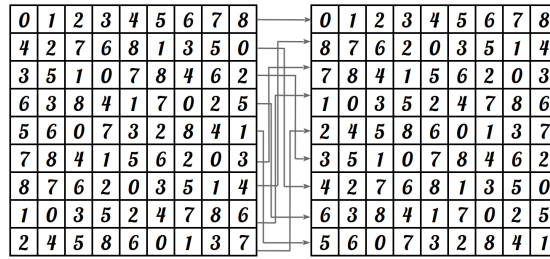


Fig. 2. Example of a pair of row-permutational ODLs

2.3 Algorithm of Search for Row-permutational ODLs

Each instance of a workunit sent to a participant of the project is defined by a mask of a DLS with pre-filled first row, main and secondary diagonals and the first element of the second row (Figure 3).

0	1	2	3	4	5	6	7	8
3	2						4	
		1					0	
			4		2			
				6				
			1		7			
		5				8		
	3						5	
7								3

Fig. 3. Example of an input file for the workunit: the mask of a DLS

The overview of the computational algorithm is as follows:

1. Launch;
2. Verify the existence of the checkpoint file and the input parameters file;
3. Set the state of the computational process basing by the content of either the checkpoint file or the input parameters file (if the former is absent);
4. Search for row-permutational DLSs.

The input data are:

- **newSquare** — a matrix of a partially filled DLS;
- **path[1][2]** — two-dimensional array of length 1, where 1 is the count of square cells to be filled, and (**path[i][0]**, **path[i][1]**) are the coordinates of a square cell to be filled according to the *i*-th position in the bypass path of the square. When bypassing the square, we can return to the same **path[i]** but with different values;
- **keyRowId**, **keyColumnId**, **keyValue** — coordinates of the “key cell” and its value indicating the end of search.

The variables of search are:

- `cellId` — the number of a cell in `path` according to which we are filling one of the square cells;
- `columns[n][n]`, `rows[n][n]`, `primary[n]`, `secondary[n]` — flag arrays indicating the use of cell values in columns, rows and diagonals;
- `cellsHistory[n][n][n]` — flag arrays indicating the previous use of cell values in the process of square filling (history of the cell values).

Bypass the matrix `newSquare` according to the path until the value in the key cell reaches the given one;

4.1. Fill the next cell of `newSquare`:

Look through the possible values of the cell, from 1 to `n`. If we managed to find such `i`-th value that has not been used in these row and column and has not been written in the history of cell values, then remember it and:

- a) Write the found value into the square;
 - 1) If the square cell already contains a value, remember it;
 - 2) Write the found value into the square matrix;
 - 3) Write the fact of using the value into the flag arrays;
 - 4) Unset the flags of using the previous value of the cell in the flag arrays, but not in the history of the cell values!
 - 5) Step forward on the bypass path of the square (`path`);
 - 6) If we have reached the path end, then the square is filled, a DLS is formed; call the row-permutational search for ODLSs for the obtained DLS.

Otherwise:

- b) Due to inability to fill the current cell with a new value, return to the previous cell:
 - 1) For the value written into the current cell, unset the flags of its usage;
 - 2) Write into the current cell an “empty value”;
 - 3) Clear the history of values for the current cell;
 - 4) Step backward on the bypass path of the square (`path`).

The algorithm of search for orthogonal diagonal mates by rows permutations does permute the rows using the similar flag arrays, and after each permutation checks the resulting square for orthogonality to the initial one.

The search can be further optimized. The initial algorithm uses ordinary C++ arrays as the flag arrays. But if one replaces them with bit arrays and performs the search for the first free element using BSF instruction called by the functions `_BitScanForward` (in case of Visual C++) and `__builtin_ffs` (in case of GNU C++), then one can remove the corresponding loops of search from the program making it faster.

The program code is available online [25].

3 RakeSearch Project

The application was implemented for Linux 32- and 64-bit, Windows 32- and 64-bit and Android ARM. The average running time of a workunit was about 5 hours for the default application and 30 minutes for the optimized application. Deadline was set to one week, the quorum was 2.

Orthogonal pairs of row-permutational DLSs have been found since the very first weeks of the project's work. Having assigned badges for their discovery, we attracted more volunteers to participate.

In December 2017, the first discovered graph of ODLs was published on the project webpage. All discovered pairs of orthogonal DLSs are being published on the project website together with the workunit name and the names of the participants whose computers found them.

At the end of April 2018, the project experienced a peak load period during the first competition of BOINC teams. In July 2018, the Formula BOINC [26] sprint was held on the basis of the project, causing the highest load period ever. In Figure 4 we provide the results receiving rate in a time interval including three days before the sprint start, the sprint itself and three days after the sprint end. Each bar on the diagram shows the number of results received during the corresponding hour. We see large spikes immediately after the race start and on the start of the last day of the competition. The results receiving rate was 10–15 times more than the average of the usual load. Such period of a high load revealed the bottleneck components of the server to be the database parameters for RAM caching, hard drive I/O performance and low parallelization of server daemons.

Basing on the results of the sprint, we optimized the MySQL instance serving the project database, and modified the algorithm of data archiving. These modifications will provide the basis for other periods of high load and improve the project scalability in general.

4 Results of the Project

At the time of June 2019, the RakeSearch project has been running for 22 months. With about 3000 active computational nodes belonging to more than 600 participants, its maximal daily average performance corresponded to the performance of a 100 TeraFLOPS cluster.

Such amount of available computational resources allowed to complete more than 23 million workunits in 22 months. The search results are represented in form of orthogonality graphs, where the vertices are DLSs of order 9, and there is an edge between two vertices if the corresponding squares are mutually orthogonal. Such graphs are undirected, without loops and multi-edges.

In total, 175 unique types of orthogonality graphs have been found in the project. Their sizes vary from 2 vertices and one edge to 649 728 vertices and 3 178 752 edges. We present 20 largest graphs in Table 1. One square per the each of graphs are presented in string representation (elements of the square are

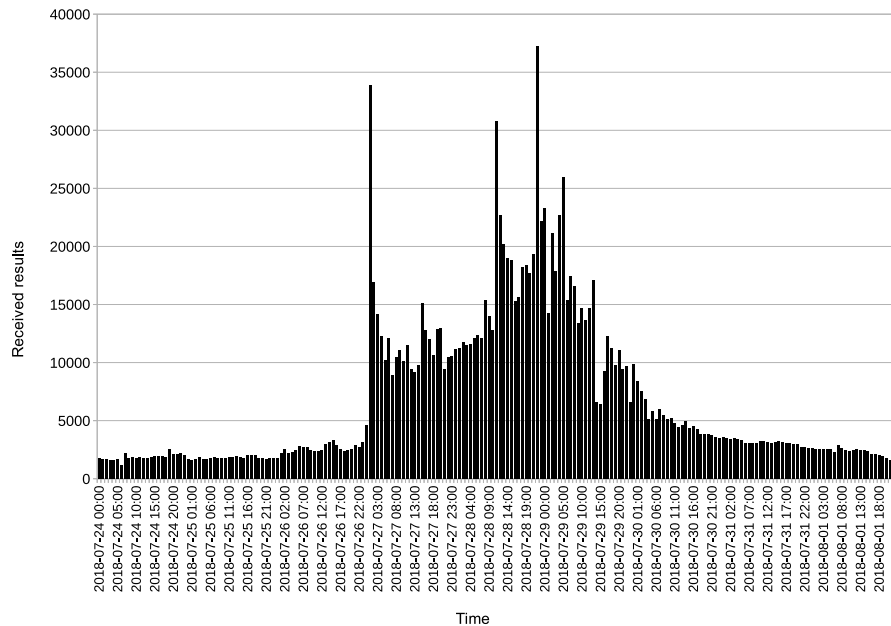


Fig. 4. Results receiving rate during the high server load

written from left to right in each row and from top to bottom row). In Figure 5, we provide the first discovered graph which has 48 vertices and 123 edges.

Some graphs that were found earlier during investigating properties of central symmetry [27] of DLSs for this moment are not present in the list which means they do not contain row-permutational square pairs. Some graphs found within the project probably may have a coinciding number of vertices and edges, however, they may not be isomorphic, which requires a separate study. All discovered graphs are visualized using the Gephi software [28] and published on the project website.

5 Conclusion

We have described the start-up of a volunteer computing project aimed at characterizing the space of diagonal Latin squares of order 9 and the experience of running it for 22 months. During this time, 175 unique types of orthogonality graphs have been found in the project. The obtained results prove the efficiency of distributed computing in unveiling the structure of the space of DLSs.

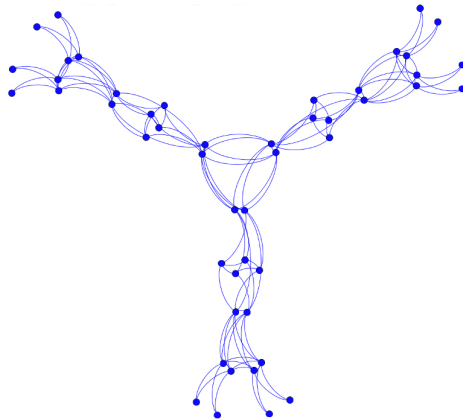


Fig. 5. The “Necklace” graph of orthogonal diagonal Latin squares of order 9

Acknowledgments

We would like to thank all volunteers who provided their computers to the project. We thank Daniel (BOINC@Poland team) for developing an optimized search application that allowed to save a lot of computational resources and time. Discussions and advice on the project forum were greatly appreciated too.

This work was supported by the Russian Foundation for Basic Research [grant numbers 18-07-00628_a, 18-37-00094_mol_a and 17-07-00317_a].

Table 1. Largest graphs of orthogonal diagonal Latin squares of order 9 obtained in the RakeSearch project.

Graph No.	Vertices	Edges	Square
1	150	1194	012345678423076815781603542648532701154860237830457126267184350506721483375218064
2	154	1263	012345678427683510561824307736450182350268741148537026604712853875106234283071465
3	164	1383	012345678127850346861703524204671853745138062456287130380526417573064281638412705
4	176	912	012345678425681730563027481137456802348570216704168523680234157851702364276813045
5	181	1321	012345678523874016271480563160532784384761205756028431847106352635217840408653127
6	208	1412	012345678628134705831207564467582310504761283785413026243076851370658142156820437
7	210	1524	012345678526814037471683250763450821305168742284537106857206314640721583138072465
8	212	1104	012345678123857046451286730637412805574063281740138562865701324208674153386520417
9	224	1112	012345678124857306453786120701624853346578012867201534285130467530462781678013245
10	292	1882	012345678423076815781653042648532701504861237835407126267184350156720483370218564
11	294	2234	012345678826057143481763025374681502540132867657208431235410786163874250708526314
12	368	1864	012345678628174305871203564463582710504861237385417026247036851730658142156720483
13	2388	7752	012345678627134805871603524380457162504861237735218046463582710248076351156720483
14	4048	6968	012345678825431760603857142451602387168274053746018235384726501237580416570163824
15	9056	45976	012345678128604537261758403873421056540863721654237810485170362307516284736082145
16	44160	231744	012345678720683514841237056473502861154760283365478102687014325238156740506821437
17	58368	546048	012345678725801346163728054538416720286053417604287135451672803870134562347560281
18	61824	374064	012345678423618750851720463780453126375861042638207514267534801504186237146072385
19	113616	675264	012345678421806537153280764508671342780453126834067215376128450645712083267534801
20	649728	3178752	012345678527186043631870524853421760745068132104237856280654317476503281368712405

References

1. C. J. Colbourn and J. H. Dinitz. *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2006.
2. N. S. Bolshakova. About one another application of Latin squares. *Vestnik of MSTU*, 8(1):170–173, 2005. in Russian.
3. P. R. Thomas and J. P. Morgan. Modern experimental design. *Statistical theory and Practice*, 1(3–4):501–506, 2007.
4. I. Anderson. *Combinatorial designs and tournaments*, volume 6. Oxford University Press, 1997.
5. J. Cooper, D. Donovan, and J. Seberry. Secret sharing schemes arising from Latin squares. *Bulletin of the Institute of Combinatorics and its Applications*, 12:33–43, 1994.
6. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North Holland Publishing Co., Jun 1977.
7. O. Zaikin, A. Zhuravlev, S. Kochemazov, and E. Vatutin. On the construction of triples of diagonal Latin squares of order 10. *Electronic Notes in Discrete Mathematics*, 54:307–312, 2016.
8. A.P. Afanasiev, I.V. Bychkov, M.O. Manzyuk, M.A. Posypkin, A.A. Semenov, and O.S. Zaikin. Technology for integrating idle computing cluster resources into volunteer computing projects. In *Proc. of The 5th International Workshop on Computer Science and Engineering, Moscow, Russia*, pages 109–114, 2015.
9. D. P. Anderson. BOINC: A platform for volunteer computing. arXiv preprint arXiv:1903.01699, 2019.
10. ET Parker. Computer investigation of orthogonal Latin squares of order ten. In *Proc. Sympos. Appl. Math*, volume 15, pages 73–81, 1963.
11. Brendan D McKay and Eric Rogoyski. Latin squares of order 10. *Electronic Journal of Combinatorics*, 2(3):1–4, 1995.
12. Brendan D McKay and Ian M Wanless. On the number of latin squares. *Annals of combinatorics*, 9(3):335–344, 2005.
13. Brendan D McKay, Alison Meynert, and Wendy Myrvold. Small Latin squares, quasigroups, and loops. *Journal of Combinatorial Designs*, 15(2):98–119, 2007.
14. Judith Egan and Ian Wanless. Enumeration of MOLS of small order. *Mathematics of Computation*, 85(298):799–824, 2016.
15. Hung-Hsuan Lin and I-Chen Wu. Solving the minimum sudoku poblem. In *2010 International Conference on Technologies and Applications of Artificial Intelligence*, pages 456–461. IEEE, 2010.
16. E. I. Vatutin, O. S. Zaikin, S. E. Kochemazov, and S. Y. Valyaev. Using volunteer computing to study some features of diagonal Latin squares. *Open Engineering*, 7(1):453–460, 2017.
17. Eduard I Vatutin, Stepan E Kochemazov, and Oleg S Zaikin. Applying volunteer and parallel computing for enumerating diagonal Latin squares of order 9. In *International Conference on Parallel Computational Technologies*, pages 114–129. Springer, 2017.
18. Eduard Vatutin, Alexey Belyshev, Stepan Kochemazov, Oleg Zaikin, and Natalia Nikitina. Enumeration of isotopy classes of diagonal latin squares of small order using volunteer computing. In *Russian Supercomputing Days*, pages 578–586. Springer, 2018.
19. O. Zaikin and S. Kochemazov. The Search for Systems of Diagonal Latin Squares Using the SAT@home Project. *International Journal of Open Information Technologies*, 3(11):4–9, 2015.

20. E. Vatutin, O. Zaikin, S. Kochemazov, and S. Valyaev. Using volunteer computing to study some features of diagonal Latin square. *Open Engineering*, 7:453–460, 2017.
21. RakeSearch. <https://rake.boincfast.ru/rakerearch>. Accessed: 2019-06-23.
22. Center for Collective Use of Karelian Research Center of the Russian Academy of Sciences. <http://cluster.krc.karelia.ru/index.php?plang=e>. Accessed: 2019-06-23.
23. N. N. Nikitina, M. O. Manzyuk, and E. I. Vatutin. Employment of distributed computing to search and explore orthogonal diagonal Latin squares of rank 9. In *Digital technologies in education, science, society: proceedings of the XI(1) All-Russian research and practice conference*, pages 97–100, Nov 2017. in Russian.
24. S. E. Bammel and J. Rothstein. The number of 9×9 Latin squares. *Discrete Mathematics*, 11(1):93–95, 1975.
25. GitHub - Nevecie/RakeSearch: Rake search of Diagonal Latin Squares. <https://github.com/Nevecie/RakeSearch>. Accessed: 2019-06-23.
26. Formula BOINC. <http://formula-boinc.org/>. Accessed: 2019-06-23.
27. E. I. Vatutin, S. E. Kochemazov, O. S. Zaikin, M. O. Manzyuk, N. N. Nikitina, and V. S. Titov. Properties of central symmetry for diagonal Latin squares (in Russian). *High-performance computing systems and technologies*, 8:74–78, 2018.
28. M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, pages 1–2, 2009.