МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2020
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2020
Proceedings of the Forty-ninth Spring Conference
of the Union of Bulgarian Mathematicians
2020

ON THE EXTENSION OF A PARTIAL SOLUTION OF d SPREADS TO A PARALLELISM*

Stela Zhelezova

A parallel algorithm for enumeration of parallelisms invariant under a predefined automorphism group is proposed. It is a parallelization of the sequential exhausted backtrack search algorithm used in [20]. The algorithm is implemented using MPI and the C++ language. It is applied for the construction of some of the parallelisms of $PG(3,4)$ possessing an automorphism group of order 2.

1. Introduction. For the basic concepts and notations concerning spreads and parallelisms of projective spaces, refer, for instance, to [8], [9] or [15].

A *t-spread* in $PG(n,q)$ is a set of distinct t-dimensional subspaces which partition the point set. A *t-parallelism* is a partition of the set of t-dimensional subspaces by t-spreads. Usually 1-spreads (1-parallelisms) are called *line spreads* (line parallelisms) or just *spreads* (parallelisms). There can be line spreads and parallelisms if n is odd.

Two parallelisms are *isomorphic* if there exists an automorphism of the projective space which maps each spread of the first parallelism to a spread of the second one.

A subgroup of the automorphism group of the projective space which maps each spread of the parallelism to a spread of the same parallelism is called *automorphism group* of the parallelism. Assuming non-trivial automorphisms is a popular approach, because the search space is reduced since the object must be a union of complete orbits of the predefined group.

The classification problem for parallelisms is the problem of determining a set of representatives for the isomorphism classes of parallelisms with definite parameters. To construct parallelisms with respect to specific additional conditions usually a backtrack search is used. Generally speaking this is a tree search problem. An internal node corresponds to a partial solution (with d spreads if the level of the node is d) and a leaf corresponds to an entire parallelism. An algorithm for the extension of a partial solution of d spreads has to explore the entire tree starting from the level d, so as to enumerate all parallelisms corresponding to the leaves.

Recently multi-processor systems have become popular. The parallelisms classification problem is well suited for a parallel computation. The extension of a partial solution to a complete solution for parallelism with given parameters can be done simultaneously

*2020 Mathematics Subject Classification: 05B25, 05B40, 05E20.
Key words: Projective space; parallelism; automorphism; parallel algorithm.
We acknowledge the provided access to the e-infrastructure of the NCHDC – part of the Bulgarian National Roadmap on RIs, with the financial support by the Grant No D01-221/03.12.2018.

on different processing devices. Development of parallel algorithms for solving combinatorial problems is of theoretical and practical significance.

Our goal is to speedup a given algorithm for combinatorial parallelisms classification by using a particular strategy of parallelization in order to achieve classification for a wider range of parameters.

There are general constructions of parallelisms. In $PG(n,2)$ they are known due to Zaicev, Zinoviev and Semakov [22] and independently by Baker [1], and in $PG(2^i-1,q)$ by Beutelspacher [6]. For all $i \geq 2$ the case $i = 2$ was proved independently by Denniston [7]. There are two infinite families of regular cyclic parallelisms in $PG(3,q)$ for $q \equiv 2(mod\ 3)$ by Penttila and Williams [12].

All parallelisms of $PG(3,2)$ are known. All parallelisms of $PG(3,3)$ were recently classified by Betten [2]. To obtain all parallelisms in other projective spaces is currently impossible. There are many classification results which rely on an assumed group of symmetries. In $PG(3,5)$ the cyclic parallelisms are classified in [13], the regular parallelisms with automorphisms of order 3 – in [19] and the parallelisms with automorphisms of order 13 – in [21].

By now $PG(3,q)$ are the most studied projective spaces and among them $PG(3,4)$ is the smallest space whose parallelisms are not classified. Parallelisms of $PG(3,4)$ have been subject of different computer-aided constructions supposing some automorphisms. All automorphisms of odd prime orders have been considered – of orders 7 [17] and 3 [20] by Topalova and Zhelezova, and 5 [18] by Topalova and Zhelezova, and independently [3] by Betten.

Some of the parallelisms with automorphisms of order 2 have been classified too, namely those which are invariant under the Baer involution [4] and the cyclic groups of order 4 [5]. The problem of the classification of the remaining parallelisms with automorphisms of order 2 is still open. We seek for approaches that will improve the given algorithms for classification of parallelisms of projective spaces possessing a particular automorphism group. One such possibility is to make algorithms faster using parallelization.

Parallel algorithms for backtrack search have been studied intensively, and in particular most of them balance the load of node explorations among the available processors [14]. Algorithms in [11] and [23] parallelize sequential backtrack search performing depth-first explorations of sub-trees locally at each processor. In [10] a deterministic algorithms for backtrack search is given which runs on an n-node mesh. In this paper we use a simple strategy to speedup a backtrack search algorithm for extending to a parallelism of a partial solution of d spreads using the work pool parallel model.

2. Sequential algorithm. Let's consider $PG(3,q)$ with $(q^4-1)/(q-1)$ points and $(q^2+1)(q^2+q+1)$ lines. There are q^2+1 lines in a spread and q^2+q+1 spreads in a parallelism. We use the method described in [20] to construct the nonisomorphic parallelisms with definite parameters invariant under a particular automorphism group.

At first the action of the chosen automorphism group on the lines of the projective space is examined. Next all the possible spread orbits are constructed in advance, and then a backtrack search is applied on them. Isomorphism testing is applied at several stages within the search. For this purpose the normalizer of the considered automorphism group is used as explained in [16].

3. Parallelization strategy. The problem under consideration has to be divided into sub-problems. If the sub-problems have a data dependence among them, a communication between the processors is needed. It can happen that the communication is more expensive than the computation itself (this depends on the particular hardware too). Splitting algorithms across many processes may result in a communication cost that causes the program to run slower than it runs in serial. So we try to divide the problem into sub-problems in such a way that there is no data dependency among them, and thus the processors do not need to communicate with each other.

For the parallel algorithm being proposed the work pool model (Fig. 1) is used.

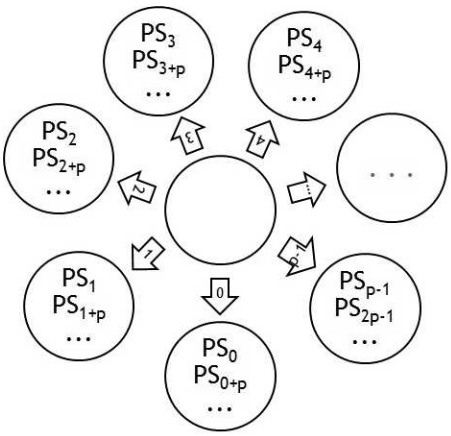

Fig. 1. A model of extension of the partial solutions

There is some additional information needed for the proper work of the algorithm, such as line and spread orbits under the action of the considered group. At first this information is read by each process. The partial solutions are enumerated and saved in advance. Next the different partial solutions are consequently assigned to different processes for balancing the load. Let p be the number of processes and m – the number of partial solutions to extend. Each process reads all partial solutions, but tries to extend only m/p of them. Distributing the work as shown in Fig. 1, the process with rank 0 will extend partial solutions with indices $0, p, 2p, \ldots$ and so on. The load-balancing depends on whether m/p is an integer, or not. If it is not, then different processes might extend a different number of partial solutions. A bigger source of imbalance, however, is the fact that different partial solutions may need a different time to be processed. After a parallelism is constructed the process writes it in its own file. At the end p files are obtained and have to be combined.

4. Some experimental results. To study the behavior of the implemented parallel algorithm we consider the construction of parallelisms of $PG(3,4)$ invariant under an automorphism group of order 2. In $PG(3,4)$ there are 85 points and 357 lines, 17 mutually disjoint lines are needed to obtain a spread, and 21 spreads make a parallelism.

We have all spreads of the projective space constructed in lexicographic order. As it is shown in [5] the Sylow subgroup of order 2 of the full automorphism group of $PG(3,4)$ has three conjugacy classes of elements of order 2. We consider an automorphism group

Table 1. Dependence of the execution time on the number of processes

p	time (sec.)	ratio
1	12284	–
4	5057	2.43
8	2573	4.77
16	1952	6.29

of order 2 which fixes 5 points and 21 lines and partitions the remaining lines in 168 orbits of length 2 ($G_{2_{21}}$). Since there are fixed lines, there must be fixed spreads too. So with respect to the action of $G_{2_{21}}$ on spreads we obtain two types of spread orbits. A spread orbit of length 1 (fixed spread) is made of some fixed lines and several whole line orbits. A spread orbit of length 2 consists of spread lines from 17 different line orbits (under $G_{2_{21}}$).

As initial data for our algorithm we construct by the sequential algorithm the first 32 partial solutions of $d = 6$ spreads in the chosen lexicographic order. They include five fixed spreads and a spread orbit of length 2. We need the line and spread orbits under $G_{2_{21}}$ too. The extension of these partial solutions by the two algorithms (sequential and parallel) lead to 80 parallelisms of $PG(3, 4)$ invariant under $G_{2_{21}}$. The comparison of the algorithm work time depending on the number of the involved processes is given in the Table 1. The results are obtained on high-performance computing system Avitohol (`http://nchdc.acad.bg/en/resources/iict/avitohol/`) which has Intel Xeon E5-2650 v2 2.6 GHz processors. The obtained data show that the considered algorithm for the extension of a partial solution of d spreads to a parallelism achieves a very good scalability.

REFERENCES

[1] R. Baker. Partitioning the planes of $AG_{2m}(2)$ into 2-designs. *Discrete Math*, **15** (1976), 205–211.

[2] A. Betten. The packings of $PG(3, 3)$. Des. Codes Cryptogr., **79**, *3* (2016), 583–595.

[3] A. Betten. Spreads and Packings – an Update. Booklet of Combinatorics 2016, Maratea (PZ), Italy, 29th May–5th June, 2016, 45.

[4] A. Betten, S. Topalova, S. Zhelezova. Parallelisms of $PG(3, 4)$ invariant under a Baer involution. Proceedings of the 16th International Workshop on Algebraic and Combinatorial Coding Theory, Svetlogorsk, Russia, 2018, 57–61.

[5] A. Betten, S. Topalova, S. Zhelezova. Parallelisms of $PG(3, 4)$ invariant under cyclic groups of order 4. In: Algebraic Informatics. CAI 2019. (eds M. Ciric, M. Droste, J. E. Pin) Lecture Notes in Computer Science, vol. **11545**, 2019, 88–99.

[6] A. Beutelspacher. On parallelisms in finite projective spaces. *Geometriae Dedicata*, **3**, *1* (1974), 35–45.

[7] R. Denniston. Packings of $PG(3, q)$. *Finite Geometric Structures and Their Applications*, Edizioni Cremonese, Rome, (1973) 193–199.

[8] J. Eisfeld, L. Storme. (Partial) t-spreads and minimal t-covers in finite projective spaces. Lecture notes from the Socrates Intensive Course on Finite Geometry and its Applications, Ghent, April 2000.

[9] N. Johnson. Combinatorics of Spreads and Parallelisms. Iowa City, CRC Press, 2010.
[10] C. Kaklamanis, G. Persiano. Branch-and-bound and backtrack search on mesh-connected arrays of processors. *Theory of Computing Systems*, **27** (1994), 471–489.
[11] K. Richard, Y. Zhang. Randomized Parallel Algorithms for Backtrack Search and Branch-and-Bound Computation. *Journal of the ACM*, **40**, *3* (1993), 765–789.
[12] T. Penttila and B. Williams, Regular packings of PG(3,q), *European Journal of Combinatorics* 19 (6) (1998) 713–720.
[13] A. Prince. The cyclic parallelisms of $PG(3,5)$. *European Journal of Combinatorics*, **19**, *5* (1998), 613–616.
[14] P. Sanders. Better algorithms for parallel backtracking. In: LNCS (eds A. Ferreira, J. D. P. Rolim), vol. **980**, 1995, 333–347
[15] L. Storme. Finite Geometry. In: The CRC Handbook of Combinatorial Designs, CRC Press, second edition, 2006, 702–729.
[16] S. Topalova. Conjugates for finding the automorphism group and isomorphism of design resolutions. *Serdica Journal of Computing*, **10**, *1* (2016), 79–92.
[17] S. Topalova, S. Zhelezova. On transitive parallelisms of $PG(3,4)$. *Appl. Algebra Engrg. Comm. Comput.*, **24**, *3–4* (2013), 159–164.
[18] S. Topalova, S. Zhelezova. On point-transitive and transitive deficiency one parallelisms of $PG(3,4)$. *Designs, Codes and Cryptography*, **75**, *1* (2015), 9–19.
[19] S. Topalova, S. Zhelezova. New Regular Parallelisms of $PG(3,5)$. *Journal of Combinatorial Designs*, **24** (2016), 473–482.
[20] S. Topalova, S. Zhelezova. New parallelisms of $PG(3,4)$. *Electronic Notes in Discrete Mathematics*, **57** (2017), 193–198.
[21] S. Topalova, S. Zhelezova. Types of spreads and duality of the parallelisms of $PG(3,5)$ with automorphisms of order 13. *Designs, Codes and Cryptography*, **87** (2019), 2–3.
[22] G. Zaicev, V. Zinoviev, N. Semakov. Interrelation of Preparata and Hamming codes and extension of Hamming codes to new double-errorcorrecting codes. Proc. of Second Intern. Symp. on Information Theory, (Armenia, USSR, 1971), Budapest, Academiai Kiado, 1973, 257–263.
[23] Y. Zhang. Parallel Algorithms for Combinatorial Search Problems. EECS Department, University of California, Berkeley Technical Report No. UCB/CSD-89-543, 1989.

Stela Zhelezova
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
p.o.box 323, 5000 Veliko Tarnovo, Bulgaria
e-mail: stela@math.bas.bg

РАЗШИРЯВАНЕ НА ЧАСТИЧНО РЕШЕНИЕ ОТ d СПРЕДА ДО ПАРАЛЕЛИЗЪМ

Стела Железова

Предложен е паралелен алгоритъм за построяване на паралелизми, инвариантни спрямо предварително зададена група от автоморфизми. Той представлява паралелизация на последователния алгоритъм за търсене с връщане, използван в New parallelisms of $PG(3,4)$ (от Топалова и Железова, в Electronic Notes in Discrete Mathematics, **57**, (2017) 193–198). Алгоритъмът е реализиран с помощта на MPI и езика C++. В статията е тествана работата му при използване на различен брой процеси и при конструиране на някои паралелизми на $PG(3,4)$, притежаващи група от автоморфизми от ред 2.