Representing Equivalence Problems for Combinatorial Objects

Iliya Bouyukliev, Mariya Dzhumalieva-Stoeva, Wolfgang Willems

Isomorphism computations take place in every classification algorithm and also in algorithms for generating objects of a certain type. In general the combinatorial classification is concerned with a given finite set of combinatorial objects A and an equivalence relation $(A, \cong)$ in it. The classification problem is to find exactly one element $a \in A$ in any equivalence class $A_i \subseteq A$, defined by the relation $\cong$. In terms of algebra the equivalence relation is defined as an action of a finite group G on the set of objects and the equivalence classes of the set are defined as orbits of the group on it. In particular two given objects are equivalent if they belong to one and the same equivalence class or one and the same orbit of G on A.

There are two general types of isomorphism problem algorithms. Let X, Y are objects in the finite set A. The first approach to check whether $X \cong Y$ is to use a specific for the certain objects algorithm, where X and Y are compared via invariants. If the values of a given invariant for both objects differ, these objects are not equivalent. Otherwise, additional computations are required to determine whether X and Y belongs to the same equivalence class. The performance of the algorithms depends at most on the order of the group G acting on the set A. In many cases G is too large and the process of equivalence search becomes very hard task. Such algorithms are developed for linear codes [4, 9, 12], combinatorial designs [10], Hadamard matrices. The second type of algorithms consists of obtaining canonical forms for both X and Y using canonical representative map. In this terms to test whether $X \cong Y$ is to test their canonical forms for equality. This approach is implemented also for linear codes [2], designs and graphs [7, 11]. In the most cases such algorithms are more effective than the specific algorithms of the first type.

In our work, we make an investigation on another type of algorithms, operating on a structure in which most types of objects can be represented. In other words, we represent the isomorphism problem of given combinatorial objects as the isomorphism problem of two basic objects - graphs and $\{0, 1\}$-matrices (binary matrices). Historically, most of the combinatorial objects are presented in terms of graph theory. Examples for representing of combinatorial objects as graphs are given by Kaski and Östergård [5] (Ch. 3) and it is proven that for many combinatorial objects, the isomorphism problem is at least as difficult as the graph isomorphism problem. Algorithms for graph isomorphism problem already exist [3]. The best known algorithm is the McKay's NAUTY [11]. On the other hand, some of the objects have more natural computer representation as binary matrices (designs, projective planes, etc). An algorithm for binary matrices isomorphism is included in the package Q-EXTENSION [1], developed by one of the authors.

It is not difficult to switch from graph isomorphism problem to binary matrix isomorphism problem as these two objects have natural representation into one another. Each combinatorial object, represented as graph, could be represented as a binary matrix too. We make representation of directed graphs, incidence structures, linear and nonlinear codes, Hadamard matrices and integer matrices directly as binary matrices and colored binary matrices, which is completely different and more efficient at least for the machine memory usage. To store an $n \times m$ binary matrix A in the computer memory, nm memory units are necessary. As each entry of A is either 0 or 1, these memory units could be bits. Moreover, the bitwise implemented algorithms have practically faster performance.

References

[1] I. Bouyukliev, What is Q-Extension?, Serdica J. Computing 1 (2007), 115–130. http://www.moi.math.bas.bg/~ iliya/Q_ext.htm

[2] I. Bouyukliev, About the code equivalence, in *Advances in Coding Theory and Cryptology*, T. Shaska, W.C. Huffman, D. Joyner, V. Ustimenko, Series on Coding Theory and Cryptology, World Scientific Publishing, Hackensack, NJ, 2007.

[3] P. Foggia, C.Sansone, M. Vento, A Performance Comparison of Five Algorithms for Graph Isomorphism, Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, Ischia, May 23-25, 2001.

[4] T. Fuelner, The automorphism grops of linear codes and canonical representatives of their semilinear isometry classes, AMC, vol.3, No.4, 2009, 363-383.

[5] P. Kaski, P. R.J. Östergård, Classification algorithms for codes and designs, Springer-Verlag, Berlin Heidelberg, 2006.

[6] P. Kaski, Algorithms for classification of combinatorial objects, Doctoral dissertation, Research report A94, Helsnki University of TEchnology, Laboratory for Tehoretical Computer Science, Espoo, Finland, June 2005.

[7] W. Kocay, On writing isomorphism programs, Computational and Constructive Design Theory (ed. W. D. Wallis), Kluwer, 1996, 135-175.

[8] D. L. Kreher and D. R. Stinson, Combinatorial Algorithms: Generation, Enumeration and Search, CRC Press, 1999.

[9] J. Leon, Computing automorphism groups of error-correcting codes, *IEEE Trans. Inform. Theory*, vol. 28, 1982, 496-511.

[10] Z. Mateva, Constructing a canonical form of a matrix in several problems about combinatorial designs, Serdica Journal of Computing, vol.2, Num 4, 2008, 349-368.

[11] B. McKay, Practical graph isomorphism, Congressus Numerantium, vol. 30, 1981, 45-87.

[12] N. Sendrier, The Support Splitting Algorithm, *IEEE Trans, Info. Theory*, vol. 46, 2000, 1193-1203.