Proceedings of the Fifteenth International Symposium on Combinatorial Search (SoCS 2022)

IPO-MAXSAT: Combining the In-Parameter-Order Strategy for Covering Array Generation with MaxSAT Solving (Extended Abstract*)

Irene Hiess, Ludwig Kampel, Michael Wagner, Dimitris E. Simos
SBA Research, MATRIS Group, Floragasse 7, 1040 Wien, Austria
ihiess@sba-research.org, lkampel@sba-research.org, mwagner@sba-research.org, dsimos@sba-research.org

Abstract

Covering arrays (CAs) are discrete objects appearing in combinatorial design theory that find practical applications, most prominently in software testing. The generation of optimized CAs is a difficult combinatorial optimization problem being subject to ongoing research. Previous studies have shown that many different algorithmic approaches are best suited for different instances of CAs. In this extended abstract we describe the IPO-MAXSAT algorithm, which adopts the prominent IPO strategy for CA generation and uses MaxSAT solving to optimize the occurring sub-problems.

Introduction

Covering arrays (CAs) are discrete objects appearing in combinatorial design theory having specific coverage properties regarding the appearance of tuples in sub-arrays. In recent years, CAs find application in a branch of automated software testing called combinatorial testing (Kuhn, Kacker, and Lei 2013). Thereby, the defining property of CAs, the coverage of all t-tuples in subarrays, has shown to be particularly beneficial when CAs are used to derive test sets, as these can reveal all interaction faults based on parameter-value combinations of up to t input parameters of the examined system, see (Kuhn et al. 2009).

A *covering array* denoted as $\mathsf{CA}(N; t, k, v)$ is defined as an $N \times k$ matrix, with entries coming from a v-ary alphabet and the property that each v-ary t-tuple appears at least once as a row of each sub-array when selecting any t columns of the array, see also (Colbourn and Dinitz 2006). The 4×3 matrix in the top left corner of Figure 1 gives an example of a $\mathsf{CA}(4; 2, 3, 2)$: selecting any two of the three columns, we find each binary 2-tuple appearing as a row. The parameter t is also called the *strength* of a CA, and we will refer to the number of rows N also as the *size* of a CA. The tuples appearing in the rows of CAs are called *t-way interactions*.

Similar to other *covering problems*, such as set cover or vertex cover, the typical problem arising with the notion of CAs is that of finding CAs with a minimal number of rows N. For given t, k and v this minimal number is called *covering array number* and denoted as $\mathsf{CAN}(t, k, v)$. CAs achieving this bound are called *optimal*.

*This extended abstract presents previously unpublished work.

Using SAT solving for CA generation, optimal arrays were found for small instances. The problem of CA existence for given CA parameters was encoded into SAT (Hnich et al. 2006; Banbara et al. 2010), such that a CA, if one exists, can be extracted from a model of the formula. In similar fashion, the optimization problem of finding an optimal CA was encoded into MaxSAT (Ansótegui et al. 2013). However, those approaches do not scale well.

In applications optimality is generally desired, but not needed, as sufficiently small CAs that are derived in a reasonable amount of time are satisfactory. In this context, several algorithms and heuristic construction techniques have been developed to generate CAs with a *small* number of rows, see (Torres-Jimenez, Izquierdo-Marquez, and Avila-George 2019) for a survey. Amongst these a very prominent family of algorithms are the so called in-parameter-order (IPO) algorithms (Lei and Tai 1998), which construct a CA by incrementally appending columns and rows to a small initial CA. Various CA generation tools, such as e.g. (Yu et al. 2013) and (Wagner et al. 2020), implement such algorithms.

In this abstract we describe IPO-MAXSAT, to the best of our knowledge, the first approach that combines MaxSAT solving with the IPO strategy for CA generation.

Introduction to In-Parameter-Order Algorithms

In (Lei and Tai 1998) the IPO strategy was introduced and initially applied for the generation of CAs of strength two. The concept was later generalized for CAs of arbitrary strength in (Lei et al. 2007).

The characteristic of the IPO strategy (see Figure 1 for a schematics) is that for the generation of a CA of strength t with k columns over a v-ary alphabet, it starts with a $v^t \times t$ array covering all t-way interactions of the first t columns. This array is then extended iteratively with one column at a time until a CA with k columns is attained. The addition of a column is called the *horizontal* extension (highlighted in blue in Figure 1). However, the addition of a column introduces several new t-way interactions, of which some might not be covered by the current array. Therefore, after each horizontal extension a *vertical* extension step (the green part in Figure 1) is performed, in which several rows can be added in order to restore coverage of all t-way interactions. Hence, after each vertical extension step we are guaranteed that the current array is a CA. Interleaving horizontal exten-

a	b	c	d	e
0	0	0	0	h_1
0	1	1	1	h_2
1	0	1	0	h_3
1	1	0	1	h_4
s_1	0	s_2	1	h_5
s_3	1	s_4	0	h_6
v_1	v_2	v_3	v_4	v_5
v_6	v_7	v_8	v_9	v_{10}

Figure 1: Schematics of the IPO strategy, for generating a binary CA of strength $t = 2$. Horizontal extension highlighted in blue, vertical extension highlighted in green and star-values highlighted in red.

Algorithm 1: IPO Strategy

Input: t, k, v
$CA \leftarrow \{0, \dots, v-1\}^t$ cross-product of first t columns
for $l \leftarrow t+1, \dots, k$ **do**
  HorizontalExtension(l)
  if there are uncovered t-way interactions **then**
    VerticalExtension(l)
  end if
end for
assigns star-values arbitrarily
return CA

sions with vertical extensions the desired $\mathsf{CA}(N; t, k, v)$ is generated. Any row that is added in a vertical extension step is initialized with so called *star-values* (also called *don't-care-values* in the literature), which represent entries that have not yet been assigned a value. These star-values are generally not considered in the horizontal extension, but enable the vertical extension to merge missing t-way interactions into existing rows. Should there remain star-values in the final $N \times k$ array, they can be assigned arbitrarily before the CA is returned. We give an overview of the IPO strategy in form of a pseudo code in Algorithm 1.

IPO-MAXSAT

Our IPO-MAXSAT algorithm is a new realization of the IPO strategy where a MaxSAT solver is utilized for horizontal extension. For every such extension a MaxSAT formula is given, i.e. we create a propositional formula in CNF, composed of hard clauses and weighted soft clauses. The array extension is then derived from the solution found by the MaxSAT solver. For vertical extension we use the greedy algorithm proposed in (Lei et al. 2007). In the following paragraph we briefly describe the MaxSAT formula for horizontal extension.

Horizontal extension In horizontal extension an existing CA is extended with a new column. Referring to Figure 1, we want to find values for the h_i in the blue part, and, unlike the IPO algorithm, also for the star-values s_i in the red parts. We aim to choose an extension, where the maximal number of t-way interactions is covered amongst all possible assignments of values for these variables. Additionally, we maximize the number of star-values in the resulting array. The formula Φ for horizontal extension consists of hard clauses for validity of assignments, soft clauses with low priority for star-value maximization and soft clauses with high priority for coverage maximization.

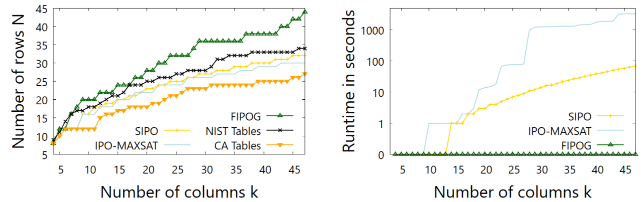

Figure 2: Experimental results for $\mathsf{CA}(N; 3, k, 2)$.

Preliminary Experiments

We conducted our initial experiments on a server with an AMD EPYC 7502P processor with 32 cores at 2.5 GHz base clock and 3.35 GHz boost clock and 128GB of RAM. For each computation we used a time limit of 3 600 seconds. In Figure 2 we present a comparison of IPO-MAXSAT using the MaxSAT solver EvalMaxSAT (Avellaneda 2021) with state-of-the-art algorithms and bounds. In particular, we compare against: **SIPO**: an algorithm implementing the IPO strategy and using Simulated Annealing to improve intermediate solutions in the horizontal extension steps (Wagner, Kampel, and Simos 2021), **FIPOG**: a representative of a state-of-the-art IPO algorithm for CA generation (Kleine and Simos 2018), **NIST Tables**: the largest online repository of CAs under (Covering Arrays Team, National Institute of Standards and Technology (NIST) 2022), generated with the IPOG-F algorithm proposed in (Forbes et al. 2008) and **CA Tables**: the currently best known upper bounds on covering array numbers (CAN) as recorded under (Colbourn 2022).

Results and Future Work

Our experimental results show that IPO-MAXSAT can produce smaller CAs when compared to similar approaches, at the cost of worse scalability. Further, our results show that even when each individual extension step is optimal, the IPO approach does not necessarily produce optimal CAs. We believe that our experiments nicely display both the possibilities and limitations of the IPO strategy and we hope that our findings can spark further research into more effective IPO algorithms.

Acknowledgments

SBA Research (SBA-K1) is a COMET Centre within the framework of COMET - Competence Centers for Excellent Technologies Programme and funded by BMK, BMDW, and the federal state of Vienna. The COMET Programme is managed by FFG.

References

Ansótegui, C.; Izquierdo, I.; Manyà, F.; and Torres Jiménez, J. 2013. A Max-SAT-Based Approach to Constructing Optimal Covering Arrays. In *Artificial Intelligence Research and Development*, 51–59. IOS Press.

Avellaneda, F. 2021. A short description of the solver EvalMaxSAT. *MaxSAT Evaluation 2021*, 10–11.

Banbara, M.; Matsunaka, H.; Tamura, N.; and Inoue, K. 2010. Generating Combinatorial Test Cases by Efficient SAT Encodings Suitable for CDCL SAT Solvers. In Fermüller, C. G.; and Voronkov, A., eds., *Logic for Programming, Artificial Intelligence, and Reasoning*, 112–126. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-16242-8.

Colbourn, C. J. 2022. Covering Array Tables for t=2,3,4,5,6. Available at http://www.public.asu.edu/~ccolbou/src/tabby/catable.html, Accessed on 2022-03-13.

Colbourn, C. J.; and Dinitz, J. H. 2006. *Handbook of combinatorial designs*. CRC press.

Covering Arrays Team, National Institute of Standards and Technology (NIST). 2022. Covering Arrays generated by IPOG-F. Available at https://math.nist.gov/coveringarrays/ipof/ipof-results.html, Accessed on 2022-03-13.

Forbes, M.; Lawrence, J.; Lei, Y.; Kacker, R. N.; and Kuhn, D. R. 2008. Refining the in-parameter-order strategy for constructing covering arrays. *Journal of Research of the National Institute of Standards and Technology*, 113(5): 287.

Hnich, B.; Prestwich, S.; Selensky, E.; and Smith, B. 2006. Constraint Models for the Covering Test Problem. *Constraints*, 11: 199–219.

Kleine, K.; and Simos, D. E. 2018. An Efficient Design and Implementation of the In-Parameter-Order Algorithm. *Mathematics in Computer Science*, 12(1): 51–67.

Kuhn, D.; Kacker, R.; and Lei, Y. 2013. *Introduction to Combinatorial Testing*. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series. Taylor & Francis.

Kuhn, R.; Kacker, R.; Lei, Y.; and Hunter, J. 2009. Combinatorial Software Testing. *Computer*, 42(8): 94–96.

Lei, Y.; Kacker, R.; Kuhn, D. R.; Okun, V.; and Lawrence, J. 2007. IPOG: A General Strategy for T-Way Software Testing. In *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)*, 549–556.

Lei, Y.; and Tai, K. 1998. In-parameter-order: a test generation strategy for pairwise testing. In *Proceedings Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No.98EX231)*, 254–261.

Torres-Jimenez, J.; Izquierdo-Marquez, I.; and Avila-George, H. 2019. Methods to Construct Uniform Covering Arrays. *IEEE Access*, 7: 42774–42797.

Wagner, M.; Kampel, L.; and Simos, D. E. 2021. Heuristically Enhanced IPO Algorithms for Covering Array Generation. In *Combinatorial Algorithms*, 571–586. Springer International Publishing.

Wagner, M.; Kleine, K.; Simos, D. E.; Kuhn, R.; and Kacker, R. 2020. CAGEN: A fast combinatorial test generation tool with support for constraints and higher-index arrays. In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 191–200.

Yu, L.; Lei, Y.; Kacker, R. N.; and Kuhn, D. R. 2013. ACTS: A Combinatorial Test Generation Tool. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, 370–375.