Constrained Covering Arrays: Resolving invalid level combination constraints

Joseph Morgan
SAS Institute Incorporated
Cary, North Carolina 27513
Joseph.Morgan@sas.com

Abstract

Covering arrays have been widely advocated as a mechanism to derive suites of test cases that ensure a pre-determined level of coverage at minimal cost [1,5]. Although these constructs have been extensively studied for over twenty years, relatively little of this effort has addressed the issue of resolving input parameter constraints [2,3]. This paper addresses that issue by extending the *replace* strategy presented in [3].

Keywords: Covering Array, Constraint Handling, Software Testing.

1. Introduction

Definition 1: A covering array **CA**(N; *t*, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$)) is an N × k array such that the *i*-th column contains v_i distinct symbols. If a **CA**(N; *t*, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$)) has the property that for any *t* coordinate projection, all $\prod v_i$ i=1,...,t, combinations of symbols exist, then it is a ***t***-covering array and is optimal if N is minimal for fixed $v_1 \cdot v_2 \cdot \ldots \cdot v_k$ and *t*.

When used for testing, columns of such arrays correspond to input parameters and the symbols in each column are levels of the parameters. Each row of the array then represents a testing scenario. Because of the covering property of these constructs, the N scenarios ensure that every *t*-way interaction is exercised and so software testing practitioners find these constructs useful. However, a particular symbol from some column may preclude specific symbols from other columns. Such situations are sometimes referred to as *invalid level combination* constraints. In [2] the authors propose a construct which extends the definition given above to account for such constraints. We present a variant of that definition here.

Definition 2: A constrained covering array **CCA**(N; *t*, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$), C) is an N × k array, with symbol set v_1, v_2,...,v_k and constraints C, such that the tuples of any *t* coordinate projection are consistent with C.

2. Resolving invalid level combination constraints

In [5] the authors describe a network interface testing problem with invalid level combination constraints. The input parameters and corresponding levels are:

1. Switch Market: CANADA, USA, UK, MEXICO, FRANCE.
2. Originating Phone Type: RES, BUS, COIN, ISDN.
3. Receiving Phone Type: RES, BUS, COIN, ISDN.
4. Originating Interface: A, B.
5. Receiving Interface: A, B.

The constraints are:

1. ISDN lines can only use Interface B.
2. RES and BUS lines can only use Interface A on the near end but can use A or B on the far end.

The approach we propose to resolve these constraints is based on a technique proposed in [3]. The technique assumes that an *unconstrained* covering array is first created and then the following steps applied.

1. **Replace step**: Identify all rows that involve invalid level combinations. For each such row, generate clones so that coverage is preserved but invalid level combinations are removed.
2. **Resolve/repair step**: For each cloned row, express constrained parameters as a *k*-partite graph (see definition below) then search the graph for maximal cliques to resolve the constraints.

To illustrate, let us say that the unconstrained array contains the following row:

Switch Mkt	Orig Phone	Recv Phone	Orig Intfc	Recv Intfc
USA	ISDN	ISDN	A	A

Since this row violates our constraints, it is replaced by the following four clones which must then be repaired:

Switch Mkt	Orig Phone	Recv Phone	Orig Intfc	Recv Intfc
USA	ISDN	ISDN	*missing*	*missing*
USA	ISDN	*missing*	*missing*	A
USA	*missing*	ISDN	A	*missing*
USA	*missing*	*missing*	A	A

In [4] the authors describe a tool that implements a branch-and-bound algorithm that exploits the connection between covering arrays and k-partite graphs to find suitable symbols for these *missing* entries.

Definition 3: Let $G = (V, E)$ denote an undirected graph where V denotes the set of vertices and E the set of edges. G is a k-partite graph if V can be partitioned into k disjoint sets $V_1, V_2, \ldots, V_k$ so that no two vertices within the same set are adjacent. That is, if $v \in V_i$ and $w \in V_j$ then $(v, w) \in E \Rightarrow i \neq j$. Furthermore, G is said to be a complete k-partite graph if all possible vertex pairs from $V_1, V_2, \ldots, V_k$ are adjacent.

Now, consider a $\mathbf{CA}(N; t, (v_1 \cdot v_2 \cdot \ldots \cdot v_k))$ and let $G = (V, E)$ be a complete k-partite graph. The k columns of the array correspond to the k disjoint sets $V_1, V_2, \ldots, V_k$ of V so that each element from v_i corresponds to a vertex in V_i. In addition, each row of the array represents a subset of V so that the subgraph of G induced by these vertices is a maximal clique.

Definition 4: Given the undirected graph G above, for some subset $Q \subseteq V$ of vertices, the induced subgraph $G(Q)$ is said to be a clique if $\forall\ v, w \in Q, v \neq w, (v, w) \in E$. If $G(Q)$ is of maximum size, we refer to it as a maximal clique.

So, by expressing the $\mathbf{CA}(N; t, (v_1 \cdot v_2 \cdot \ldots \cdot v_k))$ as a k-partite graph G, we may think of invalid level combination constraints as inducing a subgraph $G' = (V', E')$ on G where, $V_i \in V'$ iff column i participates in the constraints and $E' = E \setminus \{(v, w)$: v and w correspond to invalid level combinations$\}$.
Note: Any maximal clique of G' is a subgraph of a maximal clique of G.

Hence, resolving constraints involves finding a maximal clique in G' to replace the subgraph of the maximal clique in G that contains invalid combinations.

The induced subgraph G' for the network interface testing problem is shown below.

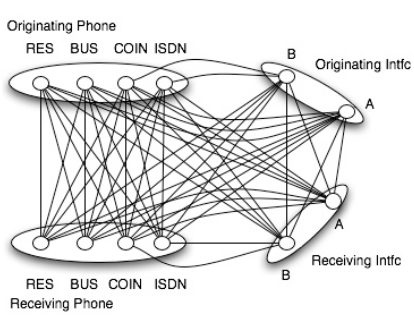

Figure 1. Constrained subgraph

Considering the first two clones, notice that for clone #1, Originating and Receiving Phone must be ISDN. Given this requirement, the search space may be pruned before attempting to find the maximal clique. So, G' may be reduced to the following graph:

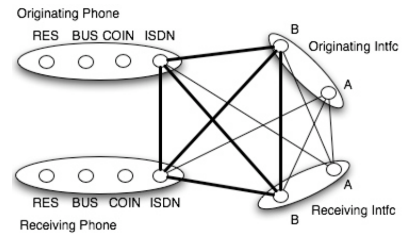

Figure 2. Constrained subgraph - clone #1

In this case, the only maximal clique is identified by the thick lines and so clone #1 may be repaired by setting Originating and Receiving Interface to B.

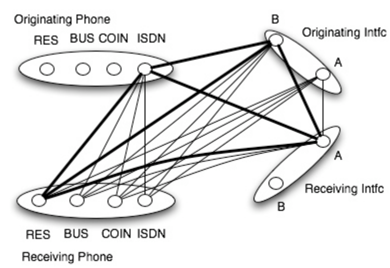

Figure 3. Constrained subgraph - clone #2

For clone #2 the reduced G' is shown above. Notice that in this case there are several possible maximal cliques. The algorithm simply selects the first one found and so clone #2 may be repaired by setting Receiving Phone Type to RES and Originating Interface to B.

References

1. D. Cohen, S. Dalal, M. Fredman, & G. Patton, "The AETG System: An approach to testing based on Combinatorial Design," IEEE Trans. Software Eng., 23(7), 437-444 , 1997.
2. M. Cohen, M. Dwyer, & J. Shi, "Interaction Testing of Highly-Configurable Systems in the Presence of Constraints," Proc. ISSTA, 129-139, 2007.
3. M. Grindal, J. Offutt, & J. Mellin, "Handling Constraints in the Input Space when Using Combination Strategies for Software Testing," School of Humanities and Informatics, University of Skovde, Tech. Rep. HS-IKI-TR-06-001, 2006.
4. J. Morgan & B. Jones, "J-FACT: A Factor-Covering Design Tool for Deriving Software Test Cases," Proc. JSM 2007, Utah.
5. A. Williams & R. Probert, "A practical strategy for testing pairwise coverage of network interfaces," Proc. 7th ISSRE, 246-254, 1996.