Scribe Notes: 11/19/09, 11/24/09

Presenter: Pingyu Zhang

Scribe: Wesley Botham

Paper: Constraint Models for the Covering Test Problem

Authors: Brahim Hnich, Steven D. Prestwich, Evgeny Selensky, Barbara M. Smith, Constraints 2006

The speaker stated that the paper

1. Addresses the covering test problem, an optimization problem.
2. Models & solves the problem as a CSP and as a SAT, and
3. Discusses the performance of the approaches adopted WRT existing systems on benchmark problems.

Regarding the CP approach, the paper

1. Describes three models for representing the covering test problem as a CSP
2. Discusses methods for reducing the search space by symmetry breaking
3. Summarizes results of experiments in solving the associated CSPs

Regarding the SAT approach, the paper

1. Describes a method for modeling the covering test problem as a SAT problem
2. Summarizes results of experiments in solving the associated SAT problem

The Covering Test Problem

The covering test problem arises from efforts to test systems that can operate in multiple configurations of its components. In order to be sure of the system's behavior under a wide variety of configurations we must perform separate experiments or simulations of the system under every configuration. Unfortunately, if we add a single new component to the system, we must repeat each test once for each allowed setting of that component. Thus the cost of such comprehensive testing grows exponentially.

To reduce the problem to a more manageable size, we relax the query. Instead of guaranteeing that the test suite (or 'covering array') contains every unique combination of all components, we only guarantee that, for each subset of a given size of the components, every unique configuration of that subset appears at least once in the covering array.

The formal statement of the covering test problem follows.

Given

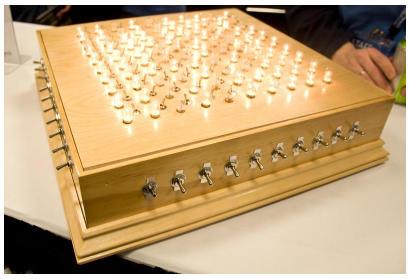
- A set of k components, each of which will be independently assigned a setting from a discrete list
- For each component, a set of g values: the 'alphabet' of settings the component can receive
- A desired covering strength t

Find

- The size b of the smallest covering array needed to ensure that every combination of t components appears in at least one test case

A covering array that covers every subset of t components in a given problem is denoted $CA(t,k,g)$ and can be represented by an array with b rows representing individual test cases and k columns listing values assumed by variables over those test cases.

Pictured is such a problem given physical form. The machine has 20 binary switches and each of the 2^{20} unique configurations of the switches maps to a potentially unique configuration of lit and unlit bulbs. In this case, $k = 20$ is the number of switches and $g = 2$ is the size of the binary alphabet $\{on, off\}$ for each switch.



Exhaustively testing every configuration of the switches implies $t = k = 20$ and requires 2^{20} tests. Relaxing the problem to $t = 3$ lowers the upper bound to only (20 choose 3)$\cdot 2^3 \approx 2^{13}$ tests. Furthermore, if the behavior we wish to detect is dependent only on the configuration of 3 switches then the smaller covering array is sufficient and the exhaustive covering array is a waste of time to generate and execute.

We can solve the covering test problem as follows: Guess a minimal value for b and attempt to construct a covering array of that size. If such an array exists, use its size as the upper bound for another guess and repeat. If such an array does not exist, use the guess as a lower bound.

Repeat until the smallest solvable b is found. It is not clear what is the complexity class of finding the covering array of minimal size, it is likely beyond NP-hard.

The CSP Models

To solve the covering test problem, we must solve the sub-problem of constructing a covering array *CA(t,k,g)* of a given size b. Three models for modeling this sub-problem as a CSP were given, respectively called the naïve model, the alternative model, and the integrated model (which combines aspects of the first two).

The Naïve Model

The most immediate CSP model is as follows.

- **Variables**:
    - First, there are variables x_{ij} representing the value of the j^{th} parameter in the i^{th} row of the covering array. The covering array has $b.k$ such elements.
    - Second, there are what we will call *auxiliary variables*. Those are Boolean variables $x_{a,cde,fhl}$ representing statements of the form "The parameters x_{ac}, x_{ad}, x_{ae} are respectively assigned the values f, h, and l." (This example is for a case where t = 3, so we have three parameters and three values associated with each Boolean variable.) There are O(bg^t(k choose t)) such variables in general. We organize them conceptually into columns where c, d, e, f, h, and l are fixed so that we can sum a column to determine whether a given instantiation of a given subset is covered by the array.
- **Domains**: The domain of each x_{ij} is its alphabet of size g. The auxiliary variables are Boolean.
- **Constraints**: Every possible instantiation of each subset of t variables must occur at least once. The scope of one such "coverage constraint" is the set of variables in one of the aforementioned columns of Booleans and there is one constraint for each such column. Thus there are O((k choose t)$\cdot g^t$) coverage constraints, each of which is a b-ary constraint.

The Alternative Model

The naïve model introduces a large number of coverage constraints and auxiliary variables. The alternative model attempts to reduce the number of constraints.

- **Variables**: We construct *compound variables* for each unique t-tuple of parameters in each row of the covering array. There will be b(k choose t) such variables.
- **Domains**: The domain of each compound variable is the set of all g^t combinations of those parameters' possible values.
- **Constraints**:
    - Conceptualizing the variables into columns listing all assignments of a compound variable in all b tests, we observe immediately that each value in the domain of a tuple must appear at least once. We also observe that it can appear at most (b-g^t +1) times to avoid squeezing out one of the other assignments. There will be (k choose t) constraints of this type.
    - We also need to ensure that assignments to these compound variables are consistent among all compound variables. This requirement is represented as a binary constraint between each pair of compound variables that lists all consistent simultaneous assignments of each. To find the number of constraints introduced here, we sum the number of pairs of compound variables with one overlapping parameter, with two overlapping parameters, etc. until t-1 overlapping parameters. This comes out to (t choose 1)·(k-t choose t-1) + (t choose 2)·(k-t choose t-2) + … + (t choose t-1)·(k-t choose 1).
    - However, many of these constraints are redundant and can be eliminated. We can first eliminate the intersection constraints between variables that share fewer than t-1 parameters because they are implicit in the (t-1)-parameter constraints. We can also observe that the intersection constraints form numerous cliques of k compound variables; in each of these cliques, all k(k-1)/2 edges can be reduced to a logically equivalent path of k-1 edges.

Note: The slides speak of compound variables sharing 'positions,' but it was agreed that it would be more accurate to speak of them sharing 'parameters.'

The Integrated Model

The naïve model introduces a large number of auxiliary variables needed for the coverage constraints to be meaningful. The alternative model replaces these auxiliary variables with a

large number of intersection constraints. The integrated model aims for the best of both worlds.

- **Variables**: Compound variables each subsuming t parameters as prescribed in the alternative model, plus the parameters themselves.
- **Domains**: As in the alternative model, the domain of each compound variable is the set of all g^t combinations of those parameters' possible values.
- **Constraints**: We eliminate all intersection constraints by introducing *channeling constraints* between the compound variables and their parameters (thus having arity t+1). The number of such constraints is (k choose t)·b, or the number of compound variables times the number of rows. Given these constraints, Generalized Arc Consistency is sufficient to deduce all information held in the intersection constraints.

Symmetry Breaking

It was noted that covering arrays are prone to having many symmetries. Those symmetries include row symmetries (we can swap any two test cases and still have a covering array), column symmetries (we can swap any two components), and value symmetries (with a binary alphabet, for example, we can swap a 1 and a 0 within a column). However, those symmetries are broken in the following cases:

- If the ordering of the tests matters in practice, we do not have row symmetry.
- If the parameters have incomparable alphabets (e.g., some are Boolean switches and others are chosen from a large list), we do not have column or value symmetry.
- If we are unwilling to test certain combinations of settings (e.g., because the box will explode), we do not have column symmetry. Brady mentioned that constraints that restrict the combinations of settings that can be tested are particularly important in real-world settings and differentiate the approaches adopted by theoreticians and software engineers. The instructor mentioned that those situations may require looking for local symmetries that need to be identified and exploited, and thus constitute an exciting topic for research).

Under ideal circumstances in which all of these symmetries apply, the size of the symmetry group is $k!\times b!\times (g!)^k$.

Experiments

Experiments were performed to test the various models, using the Ilog Solver, incorporating symmetry-breaking constraints and contrasting extensional channeling constraints (tightened by GAC) with intensional channeling constraints (tightened by bounds consistency). For the most part, the results were as predicted:

- The integrated model consistently requires fewer constraints to model a given problem and is, in practice, cheaper to solve
- Symmetry-breaking is essential for solving many larger problems
- The problem was easier to solve when modeled by intensional constraints

The SAT Model

The fourth model is a SAT model in CNF. Here we use a Boolean variable m_{ijx} for each value of each parameter in each test case so that each parameter is represented by g Booleans, exactly one of which must be true for each parameter. We also use a Boolean variable a_{ijy} for each unique assignment of each compound variable in each test case such that, again, exactly one is true for each compound variable.

Six types of constraints were described, but many of them are redundant with one another. Ultimately, the following constraints, labeled (2), (5), and (6) on the slides, are the only necessary constraints:

2. Of the m-variables for each parameter in a row, one must be false.
5. Of the a-variables for each compound variable in a row, one must be true.
6. For each a-variable, the assignment represented for the associated compound variable must be consistent with the assignments of the m-variables for the relevant parameters.

Formal proof that these constraints are necessary and sufficient was not given for brevity, although it is briefly explored in the paper.

The authors used local search to solve the SAT model. We randomly assign all variables and repeatedly flip variables in unsatisfied clauses. Of greatest interest is the heuristic used to make those local steps. Given some fixed probability value, the solver either chooses a random variable to flip or chooses a variable whose flip will break the fewest satisfied clauses and

which has been flipped the least. (This amounts to minimizing the ratio between the number of clauses broken by the flip and the total number of flips made in the past on other variables.)

Experiments indicate that local search on the SAT model scales to large problems well compared to the CSP-based approach. However, since local search does not guarantee completeness, it may be profitable to use a CSP approach to complete the solutions when local search is insufficient.

Discussion

- It was noted that Meagher and Stevens' results were not listed on many of the tests and concluded that this was likely because their results were not comparable over the problems tested.
- It was noted that local symmetries may prove worth exploring if 'dangerous' tests break global symmetries in the CSP models.