Encoding Optimal Customized Coverage Instrumentation

Peter Ohmann
ohmann@cs.wisc.edu

David Bingham Brown
bingham@cs.wisc.edu

Naveen Neelakandan
neelakandan@cs.wisc.edu

Jeff Linderoth
linderoth@wisc.edu

Ben Liblit
liblit@cs.wisc.edu

University of Wisconsin–Madison
Madison, WI, USA

ABSTRACT

Program coverage is an important software quality metric. Coverage is most commonly gathered in the testing lab during development. However, developers also sometimes use inexpensive forms of program coverage in production software. In the post-deployment scenario, users often place very strict requirements on tracing overheads and legal instrumentation strategies. This work deals specifically with optimizing program coverage instrumentation strategies given instrumentation requirements and limitations.

The problem of optimal customized coverage instrumentation is known to be NP-hard, so a polynomial-time solver is unlikely to exist. This particular report presents a fully-optimal approach to solving the problem of customized program coverage instrumentation optimization. We encode our solution as a mixed-integer linear optimization problem. We build up a mathematical model of the constraints required to satisfy required coverage instrumentation criteria, and present a complete model for solving the customized coverage instrumentation problem.

1. INTRODUCTION

This report gives the full mathematical formulation for optimal program coverage instrumentation. The report is not meant to be read in isolation. Rather, it should be read alongside the full conference paper [4], as this report glosses over many details that are fully expounded in the original paper.

Program coverage is a common metric to measure the quality of software or its test suite. However, more recently, researchers have developed technologies to gather and use coverage data in other contexts, including after initial software deployment [3, 5, 6]. Program coverage data for a particular program execution refers to traced information on what portion of a program's structure or features were exercised. For this paper, we optimize instrumentation for the binarized variants of structural coverage metrics; that is, we optimize instrumentation where each coverage probe measures the binary property of whether a program location is "covered" or "uncovered." We specifically focus on statement and edge coverage.

For a failing production run, a developer would ideally have full coverage information at the finest possible granularity. However, for many applications, full coverage tracing may impose high time or memory overhead costs. Further, developers often do not require coverage data for all program points. For example, they may focus tracing on code features (e.g., call sites [2, 3]) likely to be useful for debugging or program analysis. Further, deployed software is usually already partially tested, so developers might desire coverage only for newly-added code, or code not adequately tested before release [5, 6]. Conversely, security-sensitive code, tightly-optimized code, or code with strict real-time requirements may be off-limits for monitoring. Deployed software, therefore, demands *customized* instrumentation: coverage data for a subset of program locations satisfying limitations on legal instrumentation points.

In this paper, we begin by formally defining the customized coverage probing problem. Note that the problem is NP-hard, as proven in the accompanying full conference paper [4]. We then present an optimal solution to the problem. We walk through each of the necessary constraints for satisfying customized coverage requirements, and construct a mixed-integer linear program (MILP) whose solution identifies the optimal coverage set.

This paper is organized as follows. Section 2 formally defines the Customized Coverage Probing Problem. Section 3 walks through the full mathematical description of our solution as a mixed-integer linear optimization problem. We conclude in section 4.

2. CUSTOMIZED COVERAGE

Our goal is to determine an optimal instrumentation plan to gather *customized, binarized* program coverage information. More concretely, we are given a single procedure's control-flow graph (CFG), G; some subset of the vertices in G for which the developer desires information, D; and another subset of the vertices in G defining legal observation points, I. The problem is to determine the cheapest set of *probes* to insert into locations from I such that, for any given path p through G, the set of probes encountered along p is sufficient to determine which vertices from D were traversed along p.

The input to the problem is as follows:

- $G = (V, E)$, a directed graph with vertices V and edges E
- $e \in V$, a unique source (or entry) vertex with in-degree 0
- $I \subseteq V$, a subset of vertices that may be probed (instrumented)
- c_i, the cost of probing vertex $i \in I$, where $\forall i \in I, c_i > 0$
- $D \subseteq V$, a set of "desired" vertices that must be "covered"
- $X \subseteq V$, a set of possible termination points

2.1 Problem Definition

Definition 1 *For $i, j \in V$, $i \to j$ denotes the set of all paths from i to j in G. If $i = j$, then the trivial path (crossing no edges) is included in this set.*

Definition 2 *$V(p)$ denotes the set of all vertices encountered along path p, including the start and end vertices of p.*

Definition 3 *A set of vertices $S \subseteq I$ is called a* ***coverage set*** *of D if $\forall x \in X$ and $\forall p_1, p_2 \in e \to x$, $V(p_1) \cap S = V(p_2) \cap S \implies V(p_1) \cap D = V(p_2) \cap D$. That is, two partial executions that encounter the same S vertices must encounter the same D vertices as well. Contrapositively, paths that encounter different D vertices must be distinguishable by encountering different S vertices as well.*

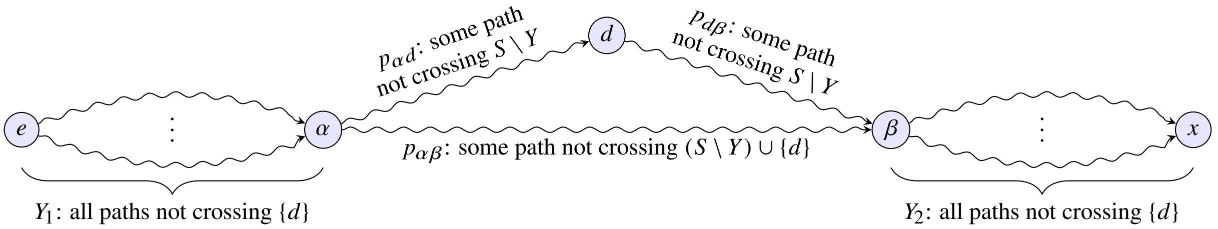
e
α
d
β
x
⋮
⋮
$p_{\alpha d}$: some path not crossing $S \setminus Y$
$p_{d\beta}$: some path not crossing $S \mid Y$
$p_{\alpha\beta}$: some path not crossing $(S \setminus Y) \cup \{d\}$
Y_1: all paths not crossing $\{d\}$
Y_2: all paths not crossing $\{d\}$

Figure 1: Pictorial representation of an ambiguous triangle

Problem Statement:

The **Customized Coverage Probing Problem** is to find a coverage set S of D such that $\sum_{s \in S} c_s$ is minimal.

3. MATHEMATICAL FORMULATION

Obtaining an optimal solution to the Customized Coverage Probing Problem is an NP-hard global optimization problem, as shown in the accompanying conference paper [4]. We formulate our optimal solution as a 0–1 mixed-integer linear optimization problem. First, we describe the sufficiency condition for a coverage set in section 3.1. In section 3.2, we provide a detailed description of how we construct the full formulation based on this condition.

3.1 Checking Sufficiency of Coverage Sets

Naïvely, from definition 3, we must examine all paths $e \to x$ (for all $x \in X$) to check if a candidate set $S \subseteq I$ is a coverage set of D. Unfortunately, for any CFG containing a cycle, there may be infinitely-many such paths. In this section, we present a simplified sufficiency check for a candidate coverage set $S \subseteq I$ that is based on locating "ambiguous triangles" between observation vertices α and β such that some $d \in D$ may or may not occur along restricted $\alpha \to \beta$ paths. S is a coverage set of D if and only if S allows no ambiguous triangles in G.

The condition is shown in fig. 1. Wavy lines represent paths crossing 0 or more edges. The core of the formulation is the three paths $p_{\alpha d}$, $p_{\alpha\beta}$, and $p_{d\beta}$. The triangle formed by these three paths corresponds to a smaller region of execution which, by its existence, demonstrates that coverage data from S is not sufficient to determine if d occurs on the path generating the coverage data. Observations (i.e., vertices from S) are usually not allowed in an ambiguous triangle, as coverage data at these observation points indicates which path through the triangle was taken. However, vertices occurring along paths from e to α (i.e., along paths in Y_1) occur before the triangle, while vertices along paths from β to some $x \in X$ (i.e., along paths in Y_2) occur after the triangle. Hence, any vertices along any path in $Y_1 \cup Y_2$ (i.e., any vertices in the set Y) may re-occur within the triangle, as they do not constitute *new observations* that provide further detail about which vertices occurred during the execution under investigation.

To formalize the above intuitions, we require one new definition:

Definition 4 ***(Connected Excluding)*** *For $\Psi \subseteq V$ and $v_1, v_2 \in V$, let $v_1 \xrightarrow{\notin\Psi} v_2$ denote the set of paths from v_1 to v_2 that* do not *cross any edges with a source or target vertex $\psi \in \Psi$.*

This definition includes trivial paths. That is, for $v \in V$, $v \xrightarrow{\notin\Psi} v$ is nonempty for any $\Psi \subseteq V$, even if $v \in \Psi$. With these definitions, for all (α, β, d) triples where

$$\alpha \in S \cup \{e\} \qquad \beta \in S \cup X \qquad d \in D \setminus S$$

we define the following sets:

$$Y_1 = e \xrightarrow{\notin\{d\}} \alpha \qquad P_{\alpha d} = \alpha \xrightarrow{\notin S\setminus Y} d$$

$$Y_2 = \bigcup_{x \in X\setminus\{d\}} \beta \xrightarrow{\notin\{d\}} x \qquad P_{\alpha\beta} = \alpha \xrightarrow{\notin(S\setminus Y)\cup\{d\}} \beta$$

$$Y = \bigcup_{\pi \in Y_1 \cup Y_2} V(\pi) \qquad P_{d\beta} = d \xrightarrow{\notin S\setminus Y} \beta$$

Then, set S is a coverage set of D if and only if:

$$Y_1 = \emptyset \ \vee Y_2 = \emptyset \ \vee P_{\alpha d} = \emptyset \ \vee P_{\alpha\beta} = \emptyset \ \vee P_{d\beta} = \emptyset$$

for all (α, β, d) triples defined above. Note that these five disjuncts correspond precisely to the five necessary parts of the ambiguous triangle pictured in fig. 1, and those necessary to form paths p_1 and p_2 from definition 3. Specifically, by selecting appropriate

$$y_1 \in Y_1 \quad y_2 \in Y_2 \quad p_{\alpha\beta} \in P_{\alpha\beta} \quad p_{\alpha d} \in P_{\alpha d} \quad p_{d\beta} \in P_{d\beta}$$

we can form the appropriate paths as

$$p_1 = y_1 \circ p_{\alpha\beta} \circ y_2$$
$$p_2 = y_1 \circ p_{\alpha d} \circ p_{d\beta} \circ y_2$$

Thus, if all five of the subpaths from the above disjunction exist for any (α, β, d) triple, then S is not a coverage set of D.

3.2 Full Formulation

In this section, we describe each piece in the construction of the full mathematical formulation of the optimization problem. The MILP itself is shown in fig. 2. We focus on some of the key "widgets" making up the different constraints in our formulation.

To begin, for notational convenience, we can define all possible ambiguous triangles for all possible sets $S \subseteq I$ as the set of triples of vertices

$$\mathcal{T} = \left\{(\alpha, \beta, d) \in (I \cup \{e\}) \times (I \cup X) \times D\right\}$$

Then, for each $(\alpha, \beta, d) \in \mathcal{T}$, we define an additional set of vertices corresponding to set Y from section 3.1. These are vertices occurring on paths from e to α or from β to a terminal vertex that do not cross vertex d:

$$Y_{\alpha\beta d} = \bigcup_{x \in X, \pi \in e \xrightarrow{\notin\{d\}} \alpha \cup \beta \xrightarrow{\notin\{d\}} x} V(\pi)$$

The sets $Y_{\alpha\beta d}$ can be constructed by simply checking basic graph connectivity, and we define the numerical parameter

$$a_{\alpha\beta di} = \begin{cases} 1 & \text{if } i \in Y_{\alpha\beta d} \\ 0 & \text{otherwise} \end{cases}$$

which is provided as input to our model.

The goal is to find S, a minimal-cost coverage set of D. We first introduce the binary selection variables

$$z_i = 1 \text{ iff } i \in S$$

to represent the selected coverage set. Next, we use five sets of binary variables, one for each path set in the characterization of a coverage set from section 3.1, that, when set to 1, will force its set of paths to be empty:

$$\begin{aligned}
s_{\alpha d} &= 1 \text{ will imply that } e \xrightarrow{\notin\{d\}} \alpha = \emptyset \\
t_{\beta d} &= 1 \text{ will imply that } \beta \xrightarrow{\notin\{d\}} x = \emptyset\ \forall x \in X \setminus \{d\} \\
u_{\alpha\beta d} &= 1 \text{ will imply that } \alpha \xrightarrow{\notin S\setminus Y_{\alpha\beta d}} d = \emptyset \\
v_{\alpha\beta d} &= 1 \text{ will imply that } \alpha \xrightarrow{\notin (S\setminus Y_{\alpha\beta d})\cup\{d\}} \beta = \emptyset \\
w_{\alpha\beta d} &= 1 \text{ will imply that } d \xrightarrow{\notin S\setminus Y_{\alpha\beta d}} \beta = \emptyset
\end{aligned}$$

Recall from section 3.1 that S is a coverage set of D if and only if at least one of these 5 sets of paths is empty for all $(\alpha, \beta, d) \in \mathcal{T}$. To force this condition, we thus introduce the constraint:

$$s_{\alpha d} + t_{\beta d} + u_{\alpha\beta d} + v_{\alpha\beta d} + w_{\alpha\beta d} \geq (1 - z_d)\ \forall(\alpha, \beta, d) \in \mathcal{T}$$

The key widget in our formulation is the ability to model, for $G = (V, E)$, whether or not there exists a path between vertices k and ℓ (i.e., is $k \rightarrow \ell \neq \emptyset$). From basic network flow theory, $k \rightarrow \ell \neq \emptyset$ if and only if the inequality system

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ij} = \begin{cases} 1 & i = k \\ 0 & i \neq k, \ell \\ -1 & i = \ell \end{cases} \quad (1)$$

$$x_{ij} \geq 0 \quad \forall(i, j) \in E \quad (2)$$

has a solution. Farkas' Lemma, or basic linear programming duality theory [1], states that this system does *not* have a solution if and only if there exist dual multipliers $\xi \in \mathbb{R}^{|V|}$ such that

$$\xi_i - \xi_j \geq 0 \quad \forall(i, j) \in E \quad (3)$$

$$\xi_k - \xi_\ell \leq -1. \quad (4)$$

Note that, in this case, we can safely bound the dual multipliers in the range $[-1, 1]$.

We use this widget as the basis of building our model. Note that we will require multipliers for many choices of starting nodes k and ending nodes ℓ. Specifically, for a fixed (α, β, d) triple, we must enforce the non-existence of one of five different sets of paths (from section 3.1), so we again define five sets of variables. These variables are associated with the existence of a particular vertex $i \in V$ along each of the five paths; thus, for each class of variables, we require one variable for each $i \in V$. For each $(\alpha, \beta, d) \in \mathcal{T}$, the following are the vertex (dual) multipliers for the linear system (3)–(4).

$\theta_i^{\alpha\beta d}$: dual multipliers associated with $e \xrightarrow{\notin\{d\}} \alpha = \emptyset$

$\eta_i^{\alpha\beta d}$: dual multipliers associated with $\beta \xrightarrow{\notin\{d\}} x = \emptyset\ \forall x \in X \setminus \{d\}$

$\pi_i^{\alpha\beta d}$: dual multipliers associated with $\alpha \xrightarrow{\notin S\setminus Y_{\alpha\beta d}} d = \emptyset$

$\mu_i^{\alpha\beta d}$: dual multipliers associated with $\alpha \xrightarrow{\notin (S\setminus Y_{\alpha\beta d})\cup\{d\}} \beta = \emptyset$

$\lambda_i^{\alpha\beta d}$: dual multipliers associated with $d \xrightarrow{\notin S\setminus Y_{\alpha\beta d}} \beta = \emptyset$

Returning to the higher-level goal, recall that each of the s, t, u, v, and w variables serves as a forcing variable, ensuring that a particular subpath from the ambiguous triangle is $\emptyset$. To take the simplest example, recall that if $s_{\alpha d} = 1$, we wish to enforce that $e \xrightarrow{\notin\{d\}} \alpha = \emptyset$. Thus, we must remove vertex d from the flow network given by (1). In the dual formulation, the equivalent operation is to remove the inequality (3) for all $(i, d) \in E$ and for all $(d, j) \in E$. Loosely, we model this constraint by removing all incoming and outgoing edges for d from G for these paths. Finally, our model should exclude $e \rightarrow \alpha$ paths only when $s_{\alpha d} = 1$ (recall: $s_{\alpha d}$ directly implies the condition $e \xrightarrow{\notin\{d\}} \alpha = \emptyset$). Thus, we need only enforce the forcing dual flow constraint (4) when $s_{\alpha d} = 1$. Algebraically, replacing the upper bound in (4) with $1 - 2s_{\alpha d}$ will serve this purpose, since, as previously stated, all dual multipliers are bounded between $[-1, 1]$. Putting this logic together gives us the dual flow system

$$\begin{aligned}
&\theta_i^{\alpha\beta d} - \theta_j^{\alpha\beta d} \geq 0 \quad \forall(\alpha, \beta, d) \in \mathcal{T}, \forall(i, j) \in E \mid i \neq d, j \neq d \\
&\theta_e^{\alpha\beta d} - \theta_\alpha^{\alpha\beta d} \leq 1 - 2s_{\alpha d} \quad \forall(\alpha, \beta, d) \in \mathcal{T}
\end{aligned}$$

The remaining dual flow systems for η, π, μ, and λ are defined in a similar fashion. However, there are some important and subtle differences. One complication arising in the definition of the variables π (which corresponds to $P_{\alpha d}$), μ (which corresponds to $P_{\alpha\beta}$), and λ (which corresponds to $P_{d\beta}$), is that we must exclude the set of vertices $S \setminus Y$ from the appropriate flow networks. Specifically, we want constraint (3) to be redundant when either $i \in S \setminus Y_{\alpha\beta d}$ or $j \in S \setminus Y_{\alpha\beta d}$. Note that $i \in S \setminus Y_{\alpha\beta d}$ is equivalent to $z_i - a_{\alpha\beta di} z_i = 1$, since z_i indicates that $i \in S$ and $a_{\alpha\beta di}$ indicates that $i \in Y_{\alpha\beta d}$. Thus, for example, in the flow system for the variables π, constraint (3) is modified to be of the form

$$\begin{aligned}
\pi_i^{\alpha\beta d} - \pi_j^{\alpha\beta d} \geq -(z_i - a_{\alpha\beta di} z_i) - (z_j - a_{\alpha\beta dj} z_j) \\
\forall(\alpha, \beta, d) \in \mathcal{T}, \forall(i, j) \in E.
\end{aligned}$$

If $t_{\beta d} = 1$, we wish to enforce that there is no path from β to any termination point that avoids passing through d. That is, $\beta \xrightarrow{\notin\{d\}} x = \emptyset$ for all $x \in X \setminus \{d\}$. To model this requirement, we introduce a special sink vertex χ with new edges (x, χ) for each $x \in X \setminus \{d\}$. Note that we *cannot* make this transformation once over the original CFG, G, since the incoming edges for χ depend on our choice of d. After the transformation, there will be at least one path in $\beta \rightarrow x$ for some $x \in X$ if and only if the expanded network has a path from $\beta \rightarrow \chi$. That is, $\bigcup_{x\in X\setminus\{d\}}(\beta \rightarrow x) \neq \emptyset$ if and only if $\beta \rightarrow \chi \neq \emptyset$. Thus, in the flow system for the variables η, constraints (3)–(4) are modified to be of the form

$$\begin{aligned}
&\eta_i^{\alpha\beta d} - \eta_j^{\alpha\beta d} \geq 0 \quad \forall(\alpha, \beta, d) \in \mathcal{T}, \forall(i, j) \in E \mid i \neq d, j \neq d \\
&\eta_\beta^{\alpha\beta d} - \eta_\chi^{\alpha\beta d} \leq 1 - 2t_{\beta d} \quad \forall(\alpha, \beta, d) \in \mathcal{T} \\
&\eta_x^{\alpha\beta d} - \eta_\chi^{\alpha\beta d} \geq 0 \quad \forall(\alpha, \beta, d) \in \mathcal{T}, \forall x \in X \mid x \neq d
\end{aligned}$$

The vertex χ is only relevant for these η constraints, and, thus, is not in V (for purposes of any other constraints).

In the end, our objective is to minimize cost

$$\sum_{i\in V} c_i z_i$$

subject to the top-level constraint

$$s_{\alpha d} + t_{\beta d} + u_{\alpha\beta d} + v_{\alpha\beta d} + w_{\alpha\beta d} \geq (1 - z_d)\ \forall(\alpha, \beta, d) \in \mathcal{T}$$

$$\min \sum_{i \in V} c_i z_i$$

subject to

$$\begin{aligned}
s_{\alpha d} + t_{\beta d} + u_{\alpha\beta d} + v_{\alpha\beta d} + w_{\alpha\beta d} &\geq 1 - z_d && \forall(\alpha, \beta, d) \in \mathcal{T} \\
\theta_i^{\alpha\beta d} - \theta_j^{\alpha\beta d} &\geq 0 && \forall(\alpha, \beta, d) \in \mathcal{T}, \forall(i, j) \in E \mid i \neq d, j \neq d \\
\theta_e^{\alpha\beta d} - \theta_\alpha^{\alpha\beta d} &\leq 1 - 2s_{\alpha d} && \forall(\alpha, \beta, d) \in \mathcal{T} \\
\eta_i^{\alpha\beta d} - \eta_j^{\alpha\beta d} &\geq 0 && \forall(\alpha, \beta, d) \in \mathcal{T}, \forall(i, j) \in E \mid i \neq d, j \neq d \\
\eta_\beta^{\alpha\beta d} - \eta_\chi^{\alpha\beta d} &\leq 1 - 2t_{\beta d} && \forall(\alpha, \beta, d) \in \mathcal{T} \\
\eta_x^{\alpha\beta d} - \eta_\chi^{\alpha\beta d} &\geq 0 && \forall(\alpha, \beta, d) \in \mathcal{T}, \forall x \in X \mid x \neq d \\
\pi_i^{\alpha\beta d} - \pi_j^{\alpha\beta d} &\geq -(z_i - a_{\alpha\beta d i} z_i) - (z_j - a_{\alpha\beta d j} z_j) && \forall(\alpha, \beta, d) \in \mathcal{T}, \forall(i, j) \in E \\
\pi_\alpha^{\alpha\beta d} - \pi_d^{\alpha\beta d} &\leq 1 - 2u_{\alpha\beta d} && \forall(\alpha, \beta, d) \in \mathcal{T} \\
\mu_i^{\alpha\beta d} - \mu_j^{\alpha\beta d} &\geq -(z_i - a_{\alpha\beta d i} z_i) - (z_j - a_{\alpha\beta d j} z_j) && \forall(\alpha, \beta, d) \in \mathcal{T}, \forall(i, j) \in E \mid i \neq d, j \neq d \\
\mu_\alpha^{\alpha\beta d} - \mu_\beta^{\alpha\beta d} &\leq 1 - 2v_{\alpha\beta d} && \forall(\alpha, \beta, d) \in \mathcal{T} \\
\lambda_i^{\alpha\beta d} - \lambda_j^{\alpha\beta d} &\geq -(z_i - a_{\alpha\beta d i} z_i) - (z_j - a_{\alpha\beta d j} z_j) && \forall(\alpha, \beta, d) \in \mathcal{T}, \forall(i, j) \in E \\
\lambda_d^{\alpha\beta d} - \lambda_\beta^{\alpha\beta d} &\leq 1 - 2w_{\alpha\beta d} && \forall(\alpha, \beta, d) \in \mathcal{T} \\
\\
z_i &\in \{0, 1\} && \forall i \in V \\
s_{\alpha d}, t_{\beta d}, u_{\alpha\beta d}, v_{\alpha\beta d}, w_{\alpha\beta d} &\in \{0, 1\} && \forall(\alpha, \beta, d) \in \mathcal{T} \\
\theta_i^{\alpha\beta d}, \pi_i^{\alpha\beta d}, \mu_i^{\alpha\beta d}, \lambda_i^{\alpha\beta d} &\in \mathbb{R} && \forall(\alpha, \beta, d) \in \mathcal{T}, \forall i \in V \\
\eta_i^{\alpha\beta d} &\in \mathbb{R} && \forall(\alpha, \beta, d) \in \mathcal{T}, \forall i \in V \cup \{\chi\}
\end{aligned}$$

Figure 2: The complete MILP formulation

which, as stated earlier, asserts that one of the 5 subpaths forming an ambiguous triangle is $\emptyset$. From section 3.1, this further implies that $S = \{i \in V$ such that $z_i = 1\}$ is a coverage set of D.

With all of the above in place, we put all constraints together, resulting in the full MILP shown in fig. 2. Note that the input is precisely that from section 2, along with the precomputed set $\mathcal{T}$, and the precomputed Y set represented by numerical parameter a. All constraints are defined over all triples in $\mathcal{T}$. From the optimal model satisfying the constraints from fig. 2 (i.e., the model that minimizes the cost function), we can then extract the optimal coverage set as $S = \{v \in I$ such that $z_v = 1\}$.

4. CONCLUSION AND FUTURE WORK

Binarized program coverage information is used in a wide variety of scenarios, from the testing lab to post-deployment monitoring. Different situations yield very different requirements for coverage, as well as different run-time overhead restrictions. We present a system that allows users to specify customized coverage criteria: desired coverage locations, as well as the set of locations that are valid for instrumentation. In this paper, we detail the constraints necessary for an instrumentation plan to satisfy these conditions. Then, we present a mixed-integer linear optimization problem whose solution yields the optimal instrumentation plan based on user-provided conditions.

While this approach is guaranteed to provide a provably-optimal result relative to a provided cost model, solving time is quite slow in practice. More details can be found in our associated full conference paper [4], where we evaluate two approaches that approximate this optimal result[1]. However, we are also examining ways to more efficiently compute an optimal result. We are investigating refinements to both the above model and the sufficiency conditions from section 3.1. In the future, research could also focus on using inexpensive analyses (including control-flow dominance and loop properties) which might speed up the expensive search for optimal coverage sets performed by MILP solvers.

5. ACKNOWLEDGMENTS

This research was supported in part by DARPA MUSE award FA8750-14-2-0270 and NSF grants CCF-0953478, CCF-1217582, CCF-1318489, and CCF-1420866. Opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

6. REFERENCES

[1] G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[1] An implementation of our optimal approach, as well as both approximations, are provided as part of our instrumenting C/C++ compiler, csi-cc. The source code is available at http://pages.cs.wisc.edu/~liblit/ase-2016-b/code/

[2] A. Nishimatsu, M. Jihira, S. Kusumoto, and K. Inoue. Call-mark slicing: an efficient and economical way of reducing slice. In *Proceedings of the 21st international conference on Software engineering*, ICSE '99, pages 422–431, New York, NY, USA, 1999. ACM. URL http://doi.acm.org/10.1145/302405.302674.

[3] P. Ohmann and B. Liblit. Lightweight control-flow instrumentation and postmortem analysis in support of debugging. In *28th International Conference on Automated Software Engineering (ASE 2013)*, Palo Alto, California, Nov. 2013. IEEE and ACM.

[4] P. Ohmann, D. B. Brown, N. Neelakandan, J. Linderoth, and B. Liblit. Optimizing customized program coverage. In *31st IEEE/ACM International Conference on Automated Software Engineering (ASE 2016)*, Singapore, Singapore, Sept. 2016. IEEE and ACM.

[5] A. Orso, D. Liang, M. J. Harrold, and R. Lipton. Gamma system: continuous evolution of software after deployment. In *Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis*, ISSTA '02, pages 65–69, New York, NY, USA, 2002. ACM. URL http://doi.acm.org/10.1145/566172.566182.

[6] C. Pavlopoulou and M. Young. Residual test coverage monitoring. In B. W. Boehm, D. Garlan, and J. Kramer, editors, *Proceedings of the 1999 International Conference on Software Engineering, ICSE' 99, Los Angeles, CA, USA, May 16-22, 1999.*, pages 277–284. ACM, 1999. URL http://portal.acm.org/citation.cfm?id=302405.302637.