Causal Inference for Statistical Fault Localization

George K. Baah
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
baah@cc.gatech.edu

Andy Podgurski
Electrical Engineering and Computer Science Dept.
Case Western Reserve University
Cleveland, OH 44106
podgurski@case.edu

Mary Jean Harrold
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
harrold@cc.gatech.edu

ABSTRACT

This paper investigates the application of *causal inference* methodology for observational studies to software fault localization based on test outcomes and profiles. This methodology combines statistical techniques for counterfactual inference with causal graphical models to obtain causal-effect estimates that are not subject to severe confounding bias. The methodology applies Pearl's Back-Door Criterion to program dependence graphs to justify a linear model for estimating the causal effect of covering a given statement on the occurrence of failures. The paper also presents the analysis of several proposed-fault localization metrics and their relationships to our causal estimator. Finally, the paper presents empirical results demonstrating that our model significantly improves the effectiveness of fault localization.

Categories and Subject Descriptors: D.2.5 [Software Engineering]: Testing and Debugging—*Diagnostics, Monitors*

General Terms: Algorithms, Experimentation

Keywords: causal inference, potential outcome model, fault localization, program analysis, debugging

1. INTRODUCTION

In recent years, there has been considerable research on using statistical, machine-learning, and data-mining techniques for software fault localization (e.g., [2, 5, 13, 14, 16]). Such techniques usually require a set of test inputs (or inputs captured in the field), a set of corresponding execution profiles, and a labeling of the test runs as passing or failing. The execution profiles typically reflect coverage of individual statements or other program elements, or they reflect the truth values of branch predicates or simple inserted predicates. The techniques assign "suspiciousness" values to these program elements to guide developers in locating faults. Using these technique, developers examine program elements in decreasing order of their suspiciousness until they discover faults. For this approach to be helpful, faulty statements must generally have higher suspiciousness values than non-faulty statements.

A simple example of a suspiciousness metric, which is used (explicitly or implicitly) in some other metrics (e.g., [12, 13, 14]), is the probability, with respect to a distribution of test inputs or operational inputs, of a program Q failing given that a certain predicate p is true. We denote this probability by $\Pr(Q \text{ fails} \mid p = \mathit{true})$, or, more concisely, by $\Pr(Q \text{ fails} \mid p)$. Predicate p indicates whether a particular kind of event occurred at a location L in Q during an execution of Q. Depending on the fault-localization technique used, p might indicate whether a particular program element (e.g., a statement or function) was covered (executed at least once), or it might indicate whether a specific branch predicate or inserted predicate evaluated to true at least once during the execution. Given a set of labeled test runs, $\Pr(Q \text{ fails} \mid p)$ is typically estimated by the sample ratio f_p/n_p, where f_p is the number of test runs for which p is true and the program fails, and where n_p is the number of test runs for which p is true. As long as the sample size n_p is not too small, the corresponding estimate should be reasonably accurate. If a predicate p is specified for each possible fault location L, then an estimate of $\Pr(Q \text{ fails} \mid p)$ can be used directly as a suspiciousness value for L.

Approaches that estimate a quantity such as the probability $\Pr(Q \text{ fails} \mid p)$ to help identify faulty statements essentially attempt to use purely statistical techniques to infer, from observational data, the causal effect of individual statements on the occurrence of program failures. However, existing statistical fault localization techniques are, by themselves, capable of revealing only statistical *associations*.[1] Valid causal inference with observational data is much more challenging than with data from randomized experiments[2] [19, 28, 30] because of the greater potential for confounding bias. *Confounding bias* [19, 22] occurs when an apparent causal effect of an event on an outcome may actually be due to an unknown confounding variable, which causes both the event and the outcome. Nevertheless, researchers must often use observational data when suitable randomized experiments are infeasible [19, 28, 30].

There is an extensive body of research on estimating causal effects from observational data, which is influential in such

ISSTA'10, July 12–16, 2010, Trento, Italy.

[1] A *statistical association* among random variables, such as *correlation*, can be defined strictly in terms of the joint distribution of the variables. A causal relationship cannot.
[2] In a *randomized experiment*, treatments are assigned randomly to individuals or units, so treatment outcomes are by design independent of the potential outcomes of the experiment [19].

diverse fields as epidemiology, economics, sociology, and machine learning (e.g., [9, 19, 22]).[3] The work of the computer scientist Judea Pearl and other researchers on the use of causal graphical models (causal graphs) [22, 29] has had an important impact on causal-inference methodology by providing both a natural way to express causal assumptions and valuable theoretical results characterizing the conditions under which valid causal inferences can be made.

Our goal in this paper is to demonstrate the relevance of causal-inference methodology for fault localization. We show that a number of suspiciousness metrics used in fault localization are related to valid estimators of the causal effects of program statements on the occurrence of program failures. These metrics suffer from confounding bias, however, because when they are used to compute the suspiciousness of a given program statement s, they do not adequately "control for" the effects of other statements on both s and the occurrence of failures. We show how this bias can be reduced or eliminated by employing causal effect estimators that do control for confounding.

There are two key aspects to applying these estimators in fault localization. The first of these is an important result by Pearl [22], which implies that confounding bias can be reduced or eliminated by conditioning on (in the sense of conditional probability) a set of variables that "blocks" all "back-door" paths (see Section 2.2.1) in a causal graph from a potential cause to the outcome variable. The second key aspect is that the correspondence between potential causal relationships among elements of a program and paths in its program-dependence graph [8] can be exploited.

In this paper, we present a simple estimator of the causal effect that covering a statement s has on program failures. The estimator is based on a linear regression model fit with data about both coverage of s and coverage of its predecessor in the control-dependence graph (if it exists). This predecessor blocks "back-door" paths under the (naive) assumption that coverage of faulty statements necessarily triggers program failures. Although this assumption does not hold in general (e.g., because failure preconditions often involve particular variable values), we present empirical results that show that ranking statements using causal-effect estimates obtained with the linear model results in rankings that are generally superior to those obtained with some non-causal (associational) suspiciousness metrics. We also discuss the issues involved in constructing a causal estimator that accounts for data dependencies and the values they carry.

The contributions of the paper are as follows:

- A new approach to fault localization based on causal-inference techniques for observational data
- A regression model for estimating the causal effect of a statement on program failures, which uses program-dependence information to control for confounding bias
- Analytical results establishing the relationships between the proposed model and other fault-localization metrics
- Empirical results indicating that the proposed model is more effective for fault localization than purely associational suspiciousness metrics, and that it is especially effective when weighted according to how often a statement is covered during failures

[3]For example, James J. Heckman won the 2000 Nobel Prize in Economics in part for his contributions in this area [9].

2. BACKGROUND

This section presents background needed to understand our new approach. Section 2.1 presents an overview of causal graphs and potential outcomes, Section 2.2 discusses the way we estimate causal effects, and Section 2.3 briefly describes program dependence graphs.

2.1 Causal Graphs and Potential Outcomes

The approach to fault localization that we present in this paper builds on Pearl's Structural Causal Model [22, 23] and on the "potential outcome" model of Neyman [21] and Rubin [27]. In this section, we sketch the elements of these frameworks that are needed in our approach. For further details about them, see References [19, 22, 23].

Causal graphs are an essential tool in Pearl's Structural Causal Model, because they are used to clearly represent both (1) the causal assumptions that permit statistical techniques to be used with observational data to make inferences about causality, not just about associations, and (2) changes such as treatments and external interventions. A *causal graph* or *causal Bayesian network* is a directed acyclic graph G whose nodes represent random variables (corresponding to causes and effects) and whose edges represent causal relationships.

An edge $X \rightarrow Y$ indicates that X (potentially) causes Y. Each random variable X has a probability distribution $P(x)$, whose form may or may not be known. (We denote the values of random variables by corresponding lowercase letters.) All causal effects associated with the *causal model* $M = (G, P)$ are *identifiable* (can be estimated) if M is *Markovian*, which means that each random variable X_i is conditionally independent of all its nondescendants, given the values of its parents (immediate predecessors) PA_i in G [22]. If M is Markovian then the joint distribution of the random variables can be factored as follows:

$$P(x_1, x_2, \ldots, x_n) = \prod_i P(x_i|pa_i) \tag{1}$$

Pearl and Verma [24] proved that M will satisfy this "parental Markov condition" if it corresponds to a *functional causal model*, in the sense that for each node X_i in G, the relationship between X_i and is parents can be described by a *structural equation* of the form

$$x_i = f_i(pa_i, u_i) \tag{2}$$

Here f_i, which represents a causal process, may be any function, and for $i = 1, \ldots, n$ the random variables U_i, which represent random errors due to unobserved variables, are *independent* of each other.[4] (If X_i has no parents, we write $x_i = f_i(u_i)$.) Put another way, M is Markovian if it represents functional relationships among a set of random variables and if any external sources of error are mutually independent. It is important to understand that, for the purposes of causal inference, the functions f_i and the distributions of the error variables U_i need not be known. In this sense, M is nonparametric. Moreover, the U_i need not be explicitly represented in G.

The *potential outcome model* or *counterfactual model* of causality has been influential in a number of fields, including economics, epidemiology, and social science [19]. Pearl

[4]Such a set of equations in itself defines a functional causal model.

has shown that his Structural Causal Model subsumes the potential outcome model, and it is convenient to use the concepts and notation of the latter, which are widely used in causal estimation, together with causal graphs. Interestingly, the potential outcome model does not explicitly represent causal relationships as in Pearl's Structural Causal Model, though it does involve states that may causally affect the outcome. For a binary cause, there are two states to which each unit (member) of the study population could be exposed, which are usually called *treatment* and *control*. A *treatment* is an intervention an investigator may apply to a set of units to assess its effects relative to no intervention (i.e., the control). These states correspond to the values of a causal exposure (treatment) variable T: for treated units, $T = 1$, and for untreated (control) units, $T = 0$. Given a measurable outcome variable Y over the study population, there are two *potential outcome random variables*, which we denote Y^1 and Y^0.[5] For a unit i, the potential outcome under the treatment state is y_i^1, and the potential outcome under the control state is y_i^0. Note that if unit i is actually treated, y_i^0 is a *counterfactual* value, because it represents the outcome that would have occurred for unit i if it *had not been treated*. Similarly, if unit i is not treated, y_i^1 is counterfactual.

The causal effect of the treatment for unit i is $\tau_i = y_i^1 - y_i^0$. It is not usually possible to calculate causal effects for individuals so average causal effects are estimated instead. The *average treatment effect* (ATE)[6] in the population is

$$\tau = E[Y^1] - E[Y^0] \tag{3}$$

where $E[\cdot]$ denotes the expectation operator.

2.2 Causal Effect Estimation

Many real-world problems, such as evaluating a medical treatment, call for estimating the average treatment effect τ in a population from a sample S of n units. Let S_1 be the subset of S consisting of the treated units, and let S_0 be the subset of S consisting of the controls. Given equation 3, a seemingly natural estimator of τ is the difference of the sample means of the outcome values for those in the treatment group and those in the control group:

$$\hat{\tau}_{re} = \frac{1}{|S_1|} \sum_{i \in S_1} y_i - \frac{1}{|S_0|} \sum_{i \in S_0} y_i \tag{4}$$

In an ideal *randomized experiment*, units would be assigned to the treatment group or the control group randomly. This assignment implies that the treatment indicator variable T is independent of the potential outcomes Y^1 and Y^0. It can be shown that in this case $\hat{\tau}_{re}$ is an *unbiased* and *consistent* estimator of the average treatment effect τ [19], meaning, respectively, that $E[\hat{\tau}_{re}] = \tau$ and that $\hat{\tau}_{re}$ converges (in probability) to τ.

In an observational study, in contrast to a randomized experiment, the treatments or policies whose effects are investigated are not under the control of the investigator(s) (e.g., because they occurred in the past). In general, treatment selection is not random (even if a random sample of population units is selected for study), and hence the treatment indicator variable T is not independent of the potential outcomes Y^1 and Y^0. Under these circumstances, the estimator $\hat{\tau}_{re}$ is likely to be biased, that is, $E[\hat{\tau}_{re}] \neq \tau$ [19]. The extent of the bias depends on the treatment selection process, which is only partly understood in most cases. In many cases, the bias will be great enough to render causal effect estimation pointless.

[5]Realized values of the potential outcomes random variables Y^1 and Y^0 are represented by lowercase letters, y^1 and y^0.
[6]In the counterfactual model, the average causal effect is referred to as the average treatment effect.

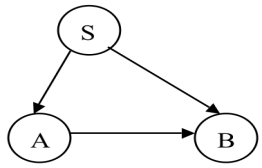

Figure 1: Causal Graph illustrating the Back-Door Criterion

2.2.1 Conditioning and The Back-Door Criterion

For an observational study, a better estimator of the average treatment effect τ is clearly needed. Although the process of treatment selection is seldom fully understood, it is often possible to characterize it in terms of one or more variables that are known or suspected of playing an important role in the selection. For example, a physician would consider the symptoms, vital signs, and medical history of a patient before selecting a treatment, and these factors can be represented by the values of a set of variables. We shall call such variables *covariates* of the treatment indicator T. If a set X of covariates accounts well for which individuals or units received the treatment and which did not, then it is possible to reduce or eliminate selection bias when estimating the average treatment effect τ on an outcome Y, by "controlling for", "conditioning on", or "adjusting on" X [19].

Pearl's *Back-Door Criterion* [22] defines, in terms of causal graphs, the characteristics that make a set X of covariates suitable for this purpose. Before presenting the Back-Door Criterion, we must first define the concepts of "blocking" and "d-separation" [22]. Note that, contrary to usual directed graph terminology, this definition refers to causal graph "paths" whose arrows (edges) may be directed either forward or backward along the path.

Definition 1. *A set S of nodes in a causal graph G is said to* ***block*** *a path p if either (1) p contains a* ***chain*** *$U \rightarrow M \rightarrow V$ or a* ***fork*** *$U \leftarrow M \rightarrow V$ whose middle node M is in S, or (2) p contains at least one* ***collider*** *$U \rightarrow M \leftarrow V$ such that the middle node M is not in S and no descendant of M is in S. If S blocks all paths from A to B, it is said to* ***d-separate*** *A and B.*

Pearl showed that if S d-separates X and Y then X and Y are *conditionally independent* given S, that is, $P(Y|X,S) = P(Y|S)$.

Definition 2. *A set S of nodes satisfies the* ***Back-Door Criterion*** *relative to a pair of nodes A, B in a causal graph G if*

1. *No node in S is a descendant of A; and*
2. *S blocks every path between A and B that contains an edge into A.*

Similarly, if A and B are two disjoint subsets of nodes in G, then S is said to satisfy the Back-Door Criterion relative to A, B if it satisfies the criterion relative to any pair $A_i \in A$, $B_j \in B$.

The name of the Back-Door Criterion comes from condition (2), which requires that paths that enter A through the "back door" be blocked. Figure 1 shows an example of a causal graph, in which there is a back-door path from A to B through S. For an intuitive explanation of d-separation and the Back-Door Criterion, see [19, 22].

The Back-Door Criterion unifies a number of strategies for how to solve treatment selection bias [19], including conditioning, stratification, and matching. If X is a set of variables that blocks all back door paths in a causal graph between a treatment indicator T and an outcome variable Y, then those paths do not contribute to the association between T and Y [19]. Hence, by conditioning on X, the average treatment effect of T upon Y can in principle be estimated without confounding bias. In practice, the causal model may be contingent, and one may not be certain that X blocks all back-door paths between T and Y. In many applications, a small degree of confounding is acceptable, and one may seek confirmatory evidence that an estimate is reasonably accurate [19].

2.2.2 Regression Estimator

There are a number of approaches to using a set X of covariates that satisfy the Back-Door Criterion, relative to a treatment indicator T and an outcome variable Y, to construct an unbiased and consistent estimator of the average treatment effect τ defined in Equation 3 [19]. In this paper, we shall employ a standard approach based on the basic statistical technique *linear regression* [4]. (More sophisticated estimators are available [19], but they are not necessary to demonstrate the relevance of causal inference to fault localization.) A linear regression model for estimating τ is

$$Y = \alpha + \tau T + \beta X + \varepsilon \tag{5}$$

In this model, α is an intercept, τ and β are coefficients[7] of T and X respectively, and ε is an error term that is uncorrelated with T. The least-squares estimate of τ, which we shall denote by $\hat{\tau}_{ls}$, is unbiased and consistent, provided the individual-level causal effect $\tau_i = y_i^1 - y_i^0$ is constant across units [19].[8]

2.3 Program Dependence Graphs

The program dependence graph [8] is a directed graph whose nodes corresponds to statements in a program and whose edges represent data dependences and control dependences between statements. Informally, node Y is *data dependent* on node X if there is a program variable v and path P from X to Y in the program's control flow graph[9] such that v is defined (assigned a value) at X, used at Y, and not redefined along P. Informally, node Y is *control dependent* on node X if X has two outgoing edges and the traversal of one edge always leads to the execution of y while the traversal of the other edge does not necessarily execute Y. Node X *dominates* node Y in a program's control flow graph if every path from the entry node to Y contains X. Node Y is *forward control dependent* on node X if Y is control dependent on X and Y does not dominate X. Intuitively, forward control dependences are control dependences that can be realized during execution without necessarily executing the dependent node more than once. Program dependence graphs can be computed statically or dynamically [18]. The dynamic version contains only dependences that are actually realized at runtime.

3. USING CAUSAL EFFECT ESTIMATES IN FAULT LOCALIZATION

We seek to demonstrate that statistical fault localization is best seen as a causal-inference problem, and that unbiased estimators of the causal effect of, say, covering a particular statement with a test are superior to biased estimators when estimates are used to rank statements for inspection by developers. How can a causal (treatment) effect estimator be used in fault localization? Given the binary outcomes from executing a set of test cases (1 denoting failure of a test and 0 denoting success) and given corresponding statement coverage profiles, we propose the following basic approach:

1. For each statement s in a faulty program Q, fit a separate linear model M_s having the form of model (5), with the following stipulations:
   (a) The outcome variable Y is 1 for a test if it fails and is 0 otherwise.
   (b) The "treatment" indicator T_s is 1 for a test if it covers s and is 0 otherwise.
   (c) If s has a forward control dependence predecessor $pred_{fcd}(s)$, then M_s has a single binary covariate C_s, which is 1 for a test if it covers $pred_{fcd}(s)$ and is 0 otherwise; if s has no forward control dependence predecessor then M_s has no covariates.
2. For each statement s, use the least-squares estimate $\hat{\tau}_{ls,s}$ of the coefficient τ_s of T_s in M_s like a suspiciousness value. That is, rank statements (for inspection by developers) in nonincreasing order of $\hat{\tau}_{ls,s}$.

Here "treatment" means executing a test that covers statement s, and "control" means executing a test that does not cover s. Thus, the linear model for statement s is

$$Y = \alpha_s + \tau_s T_s + \beta_s C_s + \varepsilon_s \tag{6}$$

The coefficient τ_s is the average treatment effect, that is, the average effect on test outcomes of covering s. Hence, it makes sense to use an unbiased estimate of τ_s such as $\hat{\tau}_{ls,s}$ as a suspiciousness value for s.

Recall from Section 2.2.2 that model (5) is intended to include a set X of covariates that block all back-door paths in the causal graph between the treatment indicator T and the outcome variable Y. Assume that (i) for program Q all causal influences of statements upon the occurrence of failure are carried by Q's data and control dependences and (ii) Y is associated with a single statement such as an output statement or the exit point of Q. Then any back-door paths from T_s to Y must begin with an edge $T_s \leftarrow pred(s)$, where $pred(s)$

[7] If X is a vector of variables then β is a vector of coefficients.
[8] If τ_i varies with i, a set of alternative assumptions must be satisfied for $\hat{\tau}_{ls}$ to be unbiased and consistent [19]
[9] A *control-flow graph* (CFG) of a program is a graph whose edges represent possible transfers of control between the program's statements.

is a predecessor of s in the program dependence graph G of Q. Under the additional assumption, which we will call the *coverage trigger assumption*, that coverage of s will necessarily trigger a failure if s is faulty, the state of s with respect to a test can be abstracted by whether or not the test covers s. Moreover, the causal relationships among Q's statements and between them and Y can be represented by the forward control dependence subgraph G_{fcd} of G. (Non-forward, loop carried control dependences are not necessary to represent control of statement coverage by branch-predicates.)

Now in reality, covering a faulty statement s may not be sufficient to trigger a failure, either because execution of s does not cause an invalid internal state or because an invalid internal state caused by s does not propagate to the program's output. To block back door paths from T_s to Y in such cases requires considering data dependences and variable values. Nevertheless, many faulty program statements are likely to trigger failures whenever they are covered, and so there is reason to think that the restricted causal-inference procedure described above will be useful for localization of those statements.

A linear model like (6) is by no means the only choice, or necessarily the best choice, for modeling the causal effect of a statement s on failures. A logistic regression model [3] is a natural alternative, and very sophisticated nonparametric and semiparametric models have been proposed for causal inference [10]. We chose a linear model to demonstrate the relevance of causal inference to fault localization because of its simplicity and the availability of fast, robust software and excellent diagnostics for linear models.

There is a further complication that should be mentioned here, involving failures due to *missing code*. In fault localization research, a fault involving missing code is by convention associated with an existing, often correct, statement near where it is believed the missing code *should* be located. Since missing code can often reasonably be placed in different locations, it may not be the case that such a statement is even moderately associated with failures.

4. ANALYSIS OF SOME FAULT LOCALIZATION METRICS

In this Section, we analyze several fault-localization metrics (suspiciousness metrics) that have been presented: Jones and Harrold's Tarantula metric [12, 13], the Ochiai metric [1], a metric called the F_1-measure, and Liblit et al.'s *Importance*(p) metric [14]. We show that each metric "embeds" in some way an estimator for the probability that a program Q fails given that statement s is covered, i.e., $\Pr(Q \text{ fails} \mid s \text{ covered})$, which we also denote by $\Pr(F \mid s)$ for brevity. This analysis will clarify the metrics' properties and relationships to each other and to the causal-effect estimator we propose. First, consider a "stripped down" version of model (6):

$$Y = \varphi_s T_s + \varepsilon_s \tag{7}$$

where we have used φ_s in place of τ_s because the coefficient is *not* the real treatment effect. This linear model omits the intercept and the variable C_s from model (6); it does *not* block back door paths between T_s and Y. Taking conditional expectations, we have

$$E[Y|T_s = 1] = Pr(F|s) = \varphi_s \tag{8}$$

Thus the least-squares estimate $\hat{\varphi}_{ls,s}$ of φ_s is an unbiased estimator of $\Pr(F \mid s)$. Clearly $\varphi_s \neq \tau_s$, and so $\hat{\varphi}_{ls,s}$ is not an unbiased estimator of the causal effect of s on failures.

In remainder of this section, we shall use the following notation to characterize a test suite used for fault localization: n is the total number of tests; n_s is the number of tests covering a particular statement s; p is the total number of tests that pass (succeed); f is the total number of tests that fail; p_s is the number of tests that pass and that also cover s; f_s is the number of test that fail and that also cover s. In the probability expressions below, we shall use P to denote the event that a test passes, F to denote the event that a test fails, and s to denote the event that a test covers statement s.

The Tarantula suspiciousness metric [12, 13] is

$$score_{Ta} = \frac{\frac{f_s}{f}}{\frac{p_s}{p} + \frac{f_s}{f}} \tag{9}$$

Multiplying the numerator f_s/f by $n/n = 1$, we obtain the ratio of the sample proportions f_s/n and f/n, which is an estimator for the probability ratio $\Pr(F \cap s)/\Pr(F) = \Pr(s|F)$. Similarly, multiplying each of the quotients p_s/p and f_s/f in the denominator of Equation (9) by n/n, we see that they are estimators for $\Pr(P \cap s)/\Pr(P) = \Pr(s|P)$ and (again) $\Pr(s|F)$, respectively. Thus, we may write

$$score_{Ta} \approx \frac{\Pr(s|F)}{\Pr(s|P) + \Pr(s|F)} \tag{10}$$

If it happens that test successes and failures are about equally likely in the set of tests used for fault localization (e.g., because it was deliberately balanced) then, since $\Pr(P) \approx \Pr(F)$), the Tarantula metric has a simpler form

$$\begin{aligned} score_{Ta} &\approx \frac{\Pr(s|F)}{\frac{\Pr(P)}{\Pr(F)}\Pr(s|P) + \Pr(s|F)} \\ &= \frac{\Pr(s|F)\Pr(F)}{\Pr(s|P)\Pr(P) + \Pr(s|F)\Pr(F)} \\ &= \frac{\Pr(s|F)\Pr(F)}{\Pr(s)} \\ &= \Pr(F|s) \end{aligned} \tag{11}$$

The last equality follows from Bayes' Theorem.[10] Thus, if test successes and failures are about equally likely, the Tarantula metric is simply an estimator for the probability of failure given that statement s is covered.

The Ochiai metric [1] is

$$score_O = \frac{f_s}{\sqrt{f(f_s + p_s)}} \tag{12}$$

Simplifying and squaring both sides of the equation, we obtain

$$(score_O)^2 = \frac{(f_s)^2}{f \cdot n_s} \tag{13}$$

Dividing numerator and denominator by n^2, we see that $(score_O)^2$ is a function of sample proportions and can be

[10]The basic form of Bayes' theorem states that

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)}$$

viewed as an estimator

$$
\begin{aligned}
(score_O)^2 &= \frac{(\frac{f_s}{n})^2}{\frac{f}{n} \cdot \frac{n_s}{n}} \\
&\approx \frac{(\Pr(F \cap s))^2}{\Pr(F)\Pr(s)} \\
&= \frac{(\Pr(s|F)\Pr(F))^2}{\Pr(F)\Pr(s)} \\
&= \Pr(s|F)\Pr(F|s)
\end{aligned}
\tag{14}
$$

(The last equality again follows from Bayes' Theorem.) Hence $score_O \approx \sqrt{\Pr(s|F)\Pr(F|s)}$. Using concepts and terminology from the field of information retrieval [17], if we view a test covering s as being analogous to a document retrieved with a given query and if we view a test that fails as being analogous to a *relevant* document, then $score_O$ approximates the square root of the product of *recall* and *precision*, where the former is $\Pr(s|F)$ and the latter is $\Pr(F|s)$. By contrast, Equation (10) indicates that the *Tarantula* metric is defined in terms of recall and $\Pr(s|P)$. The latter quantity is the *false-positive rate* for statement s (viewed as a query).

In information retrieval, a standard way of balancing recall and precision when evaluating the effectiveness of a query is to use the *F-measure* or F_1-*measure* [17], which is the (unweighted) *harmonic mean* of recall and precision. In the present context, it can be written as

$$
F_1 = \frac{2}{\frac{1}{\Pr(s|F)} + \frac{1}{\Pr(F|s)}} \tag{15}
$$

In our empirical study, we shall compare our causal effect estimator $\hat{\tau}_{ls,s}$ to F_1 and compare both to a modified version of F_1 in which $\hat{\tau}_{ls,s}$ is used as the precision term.

The fault-localization metric named *Importance* that was defined by Liblit et al. [14] is based on the F_1-measure. It applies to program predicates rather than arbitrary statements, and so it is not fully comparable to the suspiciousness metrics described above. Nevertheless, it bears some similarities to our causal effect estimator and to the modified F_1 measure mentioned above that warrant discussion, though space limitations do not permit a full analysis here. For a given predicate q, *Importance*(q) measures precision with a quantity *Increase* $\approx \Pr(F|q \text{ true}) - \Pr(F|q \text{ evaluated})$. The first term of this difference is identical to $\Pr(F|s)$ if q is true if and only if s is covered. The second term, denoted *Context*(q), is a sort of correction, intended to ensure that q is scored "not by the chance that it implies failure, but by how much difference it makes that the predicate is observed to be true versus simply reaching the line where the predicate is checked" [14]. This is somewhat like our motivation for including a coverage indicator for the forward control-dependence predecessor of a statement s in our linear model for estimating the causal effect of covering s on the outcome of a test. Note, however, that Liblit et al.'s metric has not outperformed the metrics described above [15], while our estimator does, as we shall see.

In summary, each of the proposed suspiciousness metrics we have discussed in some sense embeds an estimator for $\Pr(F|s)$, which can be viewed as the precision of s as a "failure query". Three of the metrics also embed the recall measure $\Pr(s|F)$. The significance of these facts will become more clear when we analyze the results of our empirical study.

Table 1: Subjects used for empirical studies.

Program	Vers	LOC	Tests	Description
Print-tokens	7	472	4130	lexical analyzer
Print-tokens2	10	399	4115	lexical analyzer
Replace	32	512	5542	pattern replacement
Schedule	9	292	2710	priority scheduler
Schedule2	10	301	2650	priority scheduler
Tcas	41	141	1608	altitude separation
Tot-info	23	440	1052	information measure
Sed	7	14K	363	stream editing utility
Space	38	6K	157	ADL interpreter

5. EMPIRICAL STUDIES

To evaluate the effectiveness of using the causal-effect estimator $\hat{\tau}_{ls,s}$ to compute suspiciousness scores, we implemented it and performed fault-localization studies, involving several subject programs, that compare the causal-effect estimator to other fault-localization metrics. This section first describes the empirical setup and then presents results of the empirical studies.

5.1 Empirical Setup

We extracted the control-dependence graphs and instrumented the programs using the CIL framework [20], which supports the analysis of ANSI C programs. We implemented the control-dependence algorithms in the *Objective Caml language*, because it is required for interfacing with the CIL framework. Our implementation extracts *dynamic* control-dependence graphs instead of static control-dependence graphs. Using dynamic control-dependence graphs ensures that only control dependences that are exercised at runtime appear in the control dependence graph. We implemented the causal-effect algorithm and the fault-localization metrics using R [25], which is a a system for statistical computation that consists of a language and a run-time environment.

We used the Siemens suite, Sed, and Space as subject programs in our studies. Table 1 shows their characteristics. For each subject, the first (Program), second (Vers), third (LOC), fourth (Tests), and fifth (Description) columns show the program, number of versions, lines of code, number of test cases, and a description of the program, respectively.[11] The Siemens suite consists of seven programs: Print-tokens, Print-tokens2, Replace, Schedule, Schedule2, Tcas, and Tot-info. The suite contains 132 faulty versions of these programs. However, we omitted four versions from our studies: version 32 of Replace, version 9 of Schedule2, and versions 4 and 6 of Print-tokens.We omitted these versions because (1) there were no syntactic differences between the C file of the correct version and the faulty versions of the program (e.g., because the fault was in the header file) or (2) none of the test cases failed when executed on the faulty version of the program.

The Space subject comes with 38 faulty versions and different coverage-based test suites. We randomly chose a test suite that achieves branch-coverage and executed it on Space. We used 30 faulty versions of Space. We used 30 versions of Space because, in some cases, the test suite we randomly chose had only passing or failing test cases when executed on some versions.

The Sed subject comes with seven versions with multiple

[11]We obtained the Space and Sed programs from the Software-artifact Infrastructure Repository [7].

faults per version. Each fault in a version can be activated separately. We randomly chose three different versions and activated 10 faults.

We computed the fault matrices for each version that indicates for each faulty version, which test cases pass and fail. In total, we performed the fault-localization studies on 168 faulty versions.

We instrumented each version to enable construction of the dynamic control-flow graph for each function. We then computed the dynamic control-dependence graph for each function from its dynamic control-flow graph. We also computed a statement-coverage matrix for the version. The causal-inference algorithm uses the statement-coverage matrix and the control-dependence graph as inputs.

5.2 Fault Localization

To evaluate the effectiveness of the causal-effect estimator, we compare it to $\Pr(F|s)$, Tarantula, Ochiai, and the F_1-measure. Studies 1, 2, 3, and 5 compare the causal estimator to $\Pr(F|s)$, Tarantula, Ochiai, and the F_1-measure, respectively. In Studies 4 and 7, we investigate the effectiveness of integrating the causal-effect estimator into the Ochiai and F_1-measure metrics. We also performed efficiency studies by measuring the cumulative time required to run the causal-effect estimator, $\Pr(F|s)$, Tarantula, Ochiai, and the F_1-measure.

5.2.1 Effectiveness Studies

To measure the effectiveness of the various metrics with respect to fault localization, we use a cost-measuring technique (*Cost*) used by References [2, 6, 13, 26]. The technique measures the percentage of statements a developer must examine until the faulty statement is found, assuming the statements are presented in nondecreasing order of suspiciousness and the developer starts by examining the most suspicious statement.

To compare two metrics A and B for effectiveness, we first use one of the metrics (say B) as the reference metric. We then subtract the *Cost* value for A from the *Cost* value for B. A positive value means that A performed better than B and a negative value means B performed better than A. The difference corresponds to the magnitude of improvement. For example, for a given version, if the *Cost* of A is 30% and the *Cost* of B is 40%, then the improvement of A over B is 10%, which means that developers would examine 10% fewer statements if they used A. We next present the various effectiveness studies.

Before explaining the studies, we explain the graphs we use to present the results. Figures 2, 3, 4, 5, 6, and 7 show the results for the studies. The horizontal axes represent the number of versions that show differences in the *Cost* of fault localization. The vertical axes represent the percentage difference in *Cost*s, which is the magnitude of improvement. The zero-level lines represent the performance of the reference metric. Bars above the zero-level lines represent versions for which our technique performed better than the reference metric, and bars below the zero-level line represent versions for which our technique performed worse.

Study 1: Causal-Effect Estimator vs Pr($F|s$)

The goal of this study is to compare the effectiveness for fault localization of the causal-effect estimator and the estimator f_s/n_s of $\Pr(F|s)$. To do this, we measured the *Cost* of both the causal-effect estimator and $\Pr(F|s)$ for fault localization. We used $\Pr(F|s)$ as the reference metric and subtracted the *Cost* of the causal-effect estimator from the *Cost* of $\Pr(F|s)$. Figure 2 shows the results of the study. From the 168 faulty versions, the causal-effect estimator shows improvements over $\Pr(F|s)$ in 90 versions but performs worse on two versions. The causal-effect estimator performs the same as $\Pr(F|s)$ on 78 versions. The results show that conditioning on the control-dependence predecessor reduces the bias associated with estimating the cause of a failure.

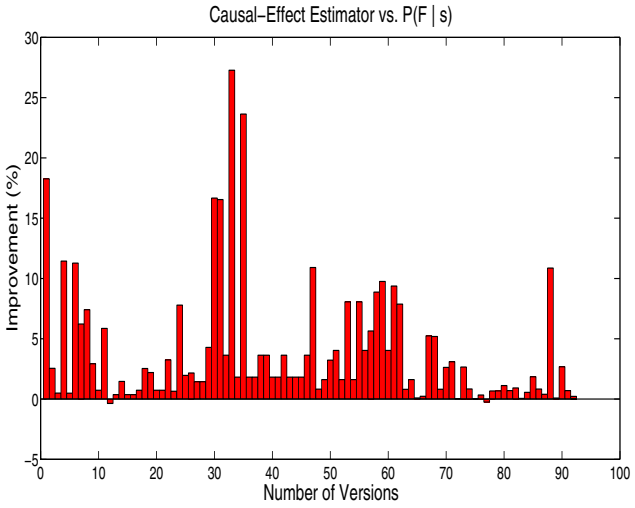

Figure 2: Comparison of Causal-Effect Estimator to $\Pr(F|s)$

Study 2: Causal-Effect Estimator vs Tarantula Metric

The goal of this study is to compare the effectiveness of the causal-effect estimator to the Tarantula metric. To do this, we used the Tarantula metric as the reference point and subtracted the *Cost* of the causal-effect estimator from the *Cost* of the Tarantula metric. Figure 3 shows the results of the study. The causal-effect estimator performs better than the Tarantula metric on 90 versions but performs worse on two versions. The causal-effect estimator performs the same as the Tarantula metric on 78 versions. The results also show that the Tarantula metric and $\Pr(F|s)$ are essentially the same. The two metrics give the same *Cost* to each faulty version, although there are differences in the suspiciousness values assigned to statements.

Study 3: Causal-Effect Estimator vs Ochiai Metric

The goal of this study is to compare the effectiveness of the causal-effect estimator to the Ochiai metric. In an empirical comparison of existing coverage-based fault-localization techniques, Abreu and colleagues [1] found the Ochiai metric to be the most effective on the set of subjects they used for the comparison. In our comparison, we computed the *Cost* of the causal-effect estimator and compared it to the *Cost* of the Ochiai metric using the Ochiai metric as the reference metric. Figure 4 shows the results of the study.

As the results show, the causal-effect estimator performs better than the Ochiai metric on 23 versions. However, it performs worse than the Ochiai metric on 70 versions. The *Cost*s of the two metrics are the same for the remaining

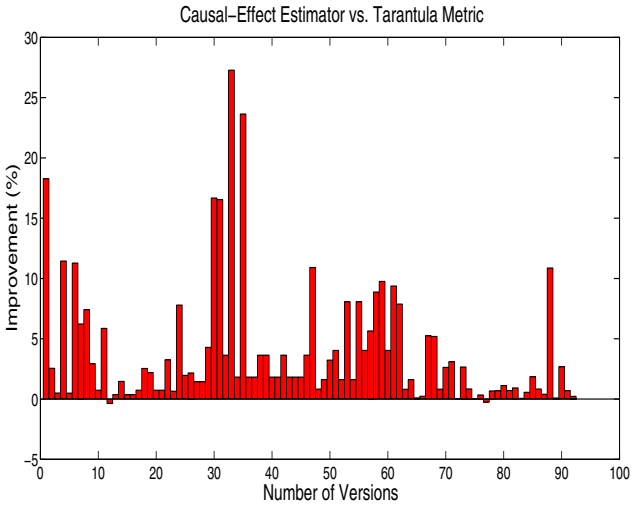

Figure 3: Comparison of Causal-Effect Estimator to *Tarantula* Metric

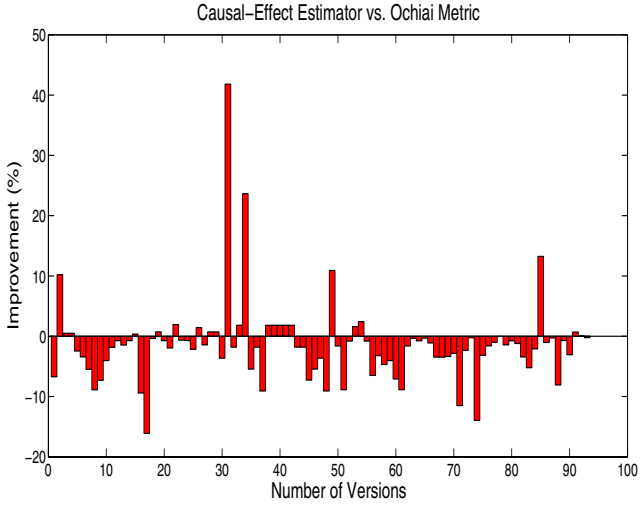

Figure 4: Comparison of Causal-Effect Estimator to Ochiai Metric

75 versions. The Ochiai metric performs better than the causal-effect estimator apparently because the Ochiai metric accounts not only for precision ($\Pr(F|s)$) but also for recall ($\Pr(s|F)$). The causal-effect estimator does not account for recall. The study shows that recall is a useful measure for fault localization. In Study 4, we examine the effectiveness of incorporating the causal-effect estimator into the Ochiai metric as the measure of precision.

Study 4: Causal-Ochiai Metric vs Ochiai Metric

The goal of this study is to compare the effectiveness of the Ochiai metric to a new metric obtained by integrating the causal-effect estimator into the Ochiai metric. We call the new metric *Causal-Ochiai*. To obtain it, we replaced the precision factor ($\Pr(F|s)$) in the Ochiai metric with the causal-effect estimator. Figure 5 shows the results of the study. Again we used the Ochiai metric as the reference metric. The results show that integrating the causal-effect estimator into the Ochiai metric improves the accuracy of

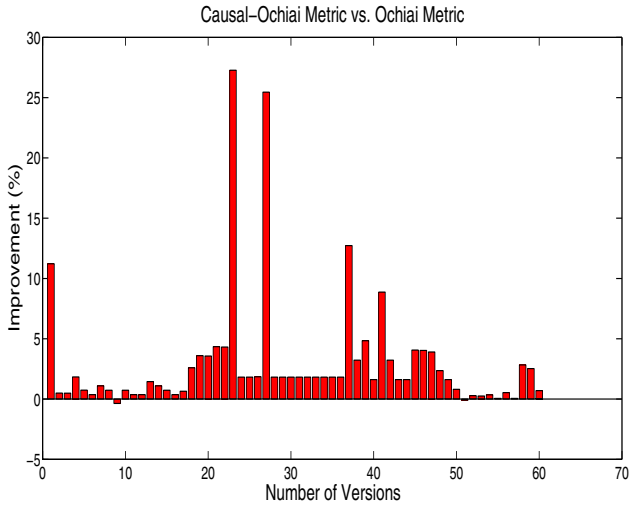

Figure 5: Comparison of Causal-Ochiai Metric to Ochiai Metric

the Ochiai metric. Out of 60 versions that showed differences in *Cost*s, the Causal-Ochiai metric performs better than the standard Ochiai metric on 58 versions but performs worse than the Ochiai metric on two versions. The Causal-Ochiai metric was therefore more effective than the standard Ochiai metric.

Study 5: Causal-Effect Estimator vs F_1-measure

The goal of this study is to compare the effectiveness of the causal-effect estimator to the F_1-measure. To do this, we measured the *Cost*s of both the causal-effect estimator and the F_1-measure for fault localization. We used the F_1-measure as the reference metric and subtracted the *Cost* of the causal-effect estimator from the *Cost* of the F_1-measure. Figure 6 shows the results of the study. The results show that out of 87 versions that show differences in *Cost*s, the causal-effect estimator performs better than the F_1-measure on 54 versions. The F_1-measure performs better than the causal-effect estimator on 33 versions. This result is surprising because the F_1-measure also incorporates a recall factor. To further investigate the behavior of the F_1-measure, we integrated the causal-effect estimator into the F_1-measure in Study 6.

Study 6: Causal F_1-measure vs F_1-measure

The goal of this study is to compare the effectiveness of the F_1-measure to a new metric, the *Causal F_1-measure*, obtained by integrating the causal-effect estimator into the F_1-measure. We obtained the Causal F_1-measure by replacing the precision factor ($\Pr(F|s)$) in the F_1-measure with the *inverse-logit*[12] of the causal-effect estimator. We use the inverse-logit function to convert the causal-effect estimate into a probability value because the F_1-measure has $precision + recall$ in its denominator. The *recall* and *precision* values are both probabilities, and thus, the causal-effect estimate must be a probability (i.e., its value must lie between 0.0 and 1.0 inclusive).

We measured the *Cost* of fault localization for both the

[12]The inverse-logit function is $invlogit(x) = exp(x)/(1 + exp(x))$.

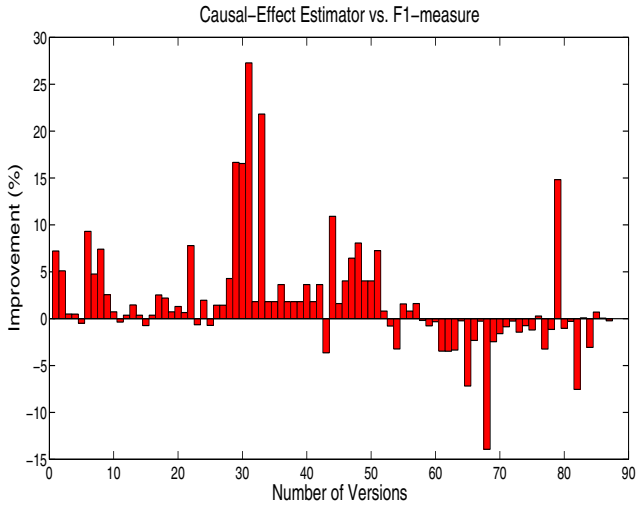

Figure 6: Comparison of Causal-Effect Estimator to F_1-measure

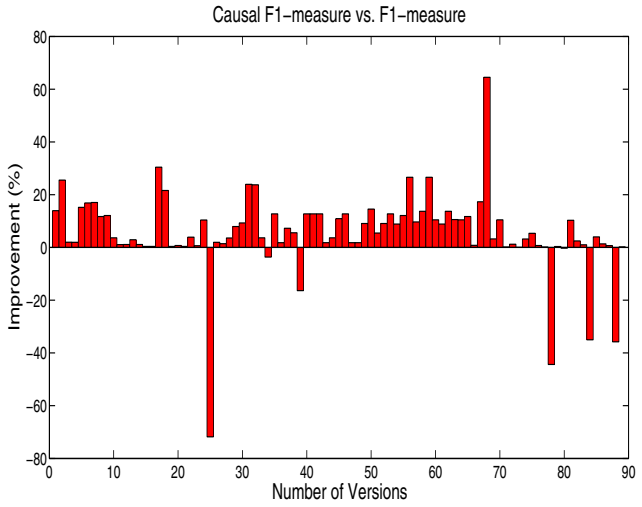

Figure 7: Comparison of Causal F_1-measure to F_1-measure

Causal F_1-measure and the F_1-measure. We used the F_1-measure as the reference metric and subtracted the *Cost* of the Causal F_1-measure from the *Cost* of the F_1-measure. Figure 7 shows the results of the study. As the results show, the Causal F_1-measure performs significantly better than the traditional F_1-measure. Out of 89 versions that show differences in *Cost*s, the Causal F_1-measure performs better on 82 versions but performs worse on seven versions. This result shows that the causal-effect estimate increases the accuracy of the standard F_1-measure.

5.2.2 Computation Time

We measured the combined computation time it took for the causal-estimator algorithm and the other fault-localization metrics to process the coverage matrices for each subject. For the 30 versions of Space, the 10 versions of Sed, and the 128 versions of the Siemens suite, the combined computation time was 12.22 minutes, 4 minutes, and 9.26 minutes, respectively. As the timings show, the algorithms are efficient. Also, the efficiency of the algorithms is dependent on the size of the coverage matrices—the larger the size of the matrices, the more time it takes for the algorithms to process the information.

6. DISCUSSION

The results of our empirical studies confirm that causal-inference techniques are relevant to fault localization and that our causal-effect estimator $\hat{\tau}_{ls,s}$ is useful in itself as a fault-localization metric and is superior to estimators of the precision measure $\Pr(F|s)$. However, our studies also indicate that the recall measure $\Pr(s|F)$ is important in fault localization and that a measure that combines the causal-effect estimator with an estimator for recall is superior to both estimators based on precision alone and estimators that combine estimators of $\Pr(F|s)$ and $\Pr(s|F)$ (see Studies 4 and 6 in Section 5.2.1).

To explain why the recall is important for fault localization, consider a statement s for which $\Pr(s|F)$ is very low. This statement will tend to be covered by few failing test cases and perhaps by few test cases overall. Thus, the sample of failing test cases available for estimating either $\Pr(F|s)$ or the causal effect of s on failures is likely to be quite small. If there are many passing test cases that cover s, the overall sample of test cases covering s will be imbalanced. If there are few passing test cases that cover s, the overall sample of test cases covering s will be small, even if it is not imbalanced. In both cases, an estimate of either $\Pr(F|s)$ or the causal effect of s is likely to be untrustworthy. The way estimates of recall are used in the Ochiai and F_1 metrics, and in their causal variants, Causal-Ochiai and Causal-F_1, addresses this issue by reducing the suspiciousness values of statements for which the recall $\Pr(s|F)$ is low and by increasing the suspiciousness values of statements for which it is high. Such weighting is likely to be much more practical than the alternative of attempting to roughly balance failures and successes covering each statement.

Although the causal-effect estimator performed well in our experiments, there are foreseeable and quite possibly common cases in which it should *not* perform well, namely faults that badly violate the coverage trigger assumption (Section 3). For example, a fault that on rare occasions causes the denominator in an expression to be zero may not be amenable to coverage-based fault localization by any means.

To address such faults, it will be necessary to develop a causal-effect estimator that controls for *all* of the dependence predecessors of a node, not just control-dependence predecessors. Such an estimator must account for the *variable values* carried by data dependencies. In some cases, it may be feasible to employ program variables directly as predictors in regression models. In other cases, it may be necessary to "bin" the values of variables. In any case, using additional, non-binary predictors is likely to necessitate employing some of the more sophisticated techniques of causal-inference methodology, such as matching/stratification and propensity scores [19].

6.1 Threats to Validity

There are three main types of validity threats that affect our studies: internal, external, and construct. Threats to internal validity concern factors that affect dependent variables without the researchers' knowledge. There is the pos-

sibility that there might be errors in our implementation—specifically, in the process of generating dynamic control-dependence graphs—that might affect the experimental results. To address potential errors in the generation of the dynamic control-dependence graphs, we compared manually generated dynamic control-dependence graphs of the functions of some of the test subjects to the dynamic control-dependence graphs we generated automatically, to ensure that the dynamic control-dependence graphs matched (which they did).

Threats to external validity occur when the results of our experiments cannot be generalized. Although we performed our empirical studies on 9 subjects with a total of 168 versions, we cannot claim that the effectiveness of causal effect estimation can be generalized to other faults in other software subjects. Also, we cannot claim to have accounted for all confounders affecting a given statement and the failure of a program. However, the results on the 168 versions demonstrate the power of causal-effect estimation as compared to non-causal suspiciousness metrics.

Threats to construct validity concern the appropriateness of the metrics used in our evaluation. It is difficult to be sure how useful human developers will find ranking metrics like those we have studied, even if their accuracy is improved considerably. More studies need to be performed to address this issue and the issues of how best to present available fault-localization results and how to integrate them with other information useful for debugging. However, the more accurate fault-localization methods are, the more meaningful such studies are likely to be.

7. RELATED WORK

The approach we presented in this paper extends our previous work on the Probabilistic Program Dependence Graph (PPDG) [2], which did not use causal-inference methodology. The PPDG is a probabilistic model of an entire program, which augments each node of a program-dependence graph with a *conditional probability table* (CPT) characterizing the conditional-probability distribution of the node's (abstract) states, given the states of its parent nodes. In the previous work, we presented a fault-localization technique that essentially uses the PPDG to rank nodes in a single failing execution trace by how anomalous their states are (given the states of their parents) compared to those from normal executions.[13] Although this proved effective, a node may have an anomalous state without being a cause of a failure. In the present work, we estimate the causal effect of covering a given statement using a regression model involving only the statement and its control dependence predecessor. The structure of dependence graphs provides the justification for using the control-dependence predecessor to control for confounding. CPTs are not needed, and the model is fit with a sample including multiple successful executions and multiple failing executions.

Several statistical fault-localization techniques [1, 13, 14, 16] have been developed that present the developer with a ranked set of statements. The techniques are similar to ours in that the techniques present the developer with a set of ranked statements to examine. The main difference is that our technique uses causal-inference methodology to find the program entity responsible for the failure whereas the other techniques attempt to find the cause of the failure using statistical-association techniques. Our empirical studies also show that our technique can significantly improve the current fault-localization techniques if the causal-effect estimate can be integrated into a technique's metric.

Cleve and Zeller [6] use Delta Debugging [31] on program states to isolate the cause of a failure in a program. Delta Debugging takes an experimental approach to causal analysis. However, Delta Debugging can be expensive because it has to ensure the consistency of memory changes and also it requires an oracle. Our approach takes an observational-study approach and, in practice, abundant observational data can easily be obtained from programs (e.g., through instrumentation).

There are also a number of fault localization techniques [11, 32] similar to Delta Debugging that rely on altering program states to find the cause of a failure. The techniques are similar to ours in that they attempt to find the cause of the failure and also they present the developer with a set of statements to examine. One difference is that our technique relies on causal analysis whereas their technique does not. Their techniques also has the same drawbacks as Delta Debugging.

8. CONCLUSION

In this paper, we have presented a novel application of causal-inference techniques to the problem of fault localization. We presented a linear estimation model for the causal effect of covering a given statement on the occurrence of failures, which is intended to eliminate or reduce confounding bias, and thereby yield better fault-localization rankings, by controlling for coverage of the statement's forward control-dependence predecessor. We justified this estimator, under certain assumptions, using Pearl's Back-Door Criterion applied to the control-dependence graph of a program, and analytically related it to several proposed fault-localization metrics. Finally, we presented the results of empirical studies indicating that the causal-effect estimator can significantly improve the effectiveness of fault localization over existing metrics. Our results suggest that established causal-inference methodology has an important place in fault-localization research. In future work, we will seek to extend our approach to address causal paths involving data dependences and causal states based on variable values.

Acknowledgements

This research was supported in part by NSF awards CCR-0306372, SBE-0123532, and CCF-0725202, an award from Tata Consultancy Services, and an IBM Software Quality Innovation Award to Georgia Tech, by NSF awards CCF-0820217 and CCF 0702693 and by an award from ABB Corporation to Case Western Reserve University, and by a FACES (Facilitating Academic Careers in Engineering and Science) fellowship award to George Baah. The anonymous reviewers provided many helpful suggestions that improved the presentation of the paper.

9. REFERENCES

[1] R. Abreu, P. Zoeteweij, and A. J. C. van Gemund. On the Accuracy of Spectrum-based Fault Localization. In

[13] In principle, the PPDG could be used to compute the probabilities of arbitrary node-state configurations, but this could be prohibitively expensive in terms of computation time and space and the amount of data needed to accurately estimate all CPTs.

Proceedings of the Testing: Academic and Industrial Conference Practice and Research Techniques, pages 89–98, 2007.

[2] G. K. Baah, A. Podgurski, and M. J. Harrold. The Probabilistic Program Dependence Graph and Its Application to Fault Diagnosis. In *Proceedings of International Symposium for Software Testing and Analysis*, July 2008.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] G. Casella and R. L. Berger. *Statistical Inference*. Thomson Learning, 2002.

[5] H. Cheng, D. Lo, Y. Zhou, X. Wang, and X. Yan. Identifying Bug Signatures Using Discriminative Graph Mining. In *Proceedings of the International Symposium on Software Testing and Analysis*, July 2009.

[6] H. Cleve and A. Zeller. Locating Causes of Program Failures. In *Proceedings of the International Symposium on the Foundations of Software Engineering*, pages 342–351, May 2005.

[7] H. Do, S. Elbaum, and G. Rothermel. Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact. *Empirical Software Engineering*, 10(4):405–435, 2005.

[8] J. Ferrante, K. J. Ottenstein, and J. D. Warren. The Program Dependence Graph and Its Use in Optimization. *ACM Transactions on Programming Languages and Systems*, 9(3):319–349, July 1987.

[9] J. J. Heckman. Microdata, Heterogeneity and the Evaluation of Public Policy. *Nobel Lectures*, Economics 1996-2000:255–322, 2000.

[10] G. W. Imbens. Nonparametric Estimation of Average Treatment Effects Under Exogeneity: A Review. *Review of Economics and Statistics*, 86(1):4–29, 2004.

[11] D. Jeffrey, N. Gupta, and R. Gupta. Fault Localization Using Value Replacement. In *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, pages 167–178, New York, NY, USA, 2008. ACM.

[12] J. Jones and M. J. Harrold. Empirical Evaluation of the Tarantula Automatic Fault-Localization Technique. In *Proceedings of the International Conference on Automated Software Engineering*, pages 273–282, November 2005.

[13] J. Jones, M. J. Harrold, and J. Stasko. Visualization of Test Information to Assist Fault Localization. In *Proceedings of the International Conference on Software Engineering*, pages 467–477, May 2002.

[14] B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan. Scalable Statistical Bug Isolation. In *Proceedings of the Conference on Programming Language Design and Implementation*, pages 15–26, June 2005.

[15] C. Liu, L. Fei, X. Yan, J. Han, and S. Midkiff. Statistical Debugging: A Hypothesis Testing-Based Approach. *IEEE Transactions on Software Engineering*, 32:841–848, 2006.

[16] C. Liu, X. Yan, L. Fei, J. Han, and S. P. Midkiff. SOBER: Statistical Model-based Bug Localization. In *Proceedings of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 286–295, September 2005.

[17] C. D. Manning, Prabhakar, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[18] W. Masri and A. Podgurski. Algorithms and Tool Support for Dynamic Information Flow Analysis. *Information and Software Technology*, 51(2):385–404, 2009.

[19] S. L. Morgan and C. Winship. *Counterfactuals and Causal Inference: Methods and Principles of Social Research*. Cambridge University Press, 2007.

[20] G. C. Necula, S. McPeak, S. P. Rahul, and W. Weimer. CIL: Intermediate Language and Tools for Analysis and Transformation of C Programs. In *Proceedings of the International Conference on Compiler Construction*, pages 213–228, April 2002.

[21] J. S. Neyman. On the Application of Probability Theory to Agricultural Experiments. Essay on Principles. *Statistical Science*, 5:465–480, 1923.

[22] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, San Francisco, CA, USA, 2000.

[23] J. Pearl. An Introduction to Causal Inference. Technical report, UCLA Cognitive Systems Laboratory, 2009.

[24] J. Pearl and T. Verma. A Theory of Inferred Causation. *In J.A Allen, R. Fikes, and E. Sandewall (Eds.), Principles of Knowledge Representation and Reasoning: Proceeding of the 2nd International Conference*, pages 441–452, 1991.

[25] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008.

[26] M. Renieris and S. Reiss. Fault Localization With Nearest Neighbor Queries. In *International Conference on Automated Software Engineering*, pages 30–39, November 2003.

[27] D. Rubin. Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies. *Journal of Educational Psychology*, 66:688–701, 1974.

[28] D. B. Rubin. The Design versus the Analysis of Observational Studies for Causal Effects: Parallels With the Design of Randomized Trials. In *Statistics in Medicine*, 2006.

[29] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search, 2nd Edition*. The MIT Press, December 2001.

[30] C. Winship and S. L. Morgan. The Estimation of Causal Effects from Observational Data. *Annual Review of Sociology*, 25:659–707, 1999.

[31] A. Zeller. Isolating cause-effect chains from computer programs. In *Proceedings ACM SIGSOFT 10th International Symposium on the Foundations of Software Engineering*, November 2002.

[32] X. Zhang, R. Gupta, and N. Gupta. Locating faults through automated predicate switching. In *Proceedings of the 28th International Conference on Software Engineering*, May 2006.