Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2016-003 February 18, 2016

An Analysis of the Search Spaces for Generate and Validate Patch Generation Systems

Fan Long and Martin Rinard

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

An Analysis of the Search Spaces for Generate and Validate Patch Generation Systems

Fan Long and Martin Rinard
MIT EECS & CSAIL, USA
{fanl, rinard}@csail.mit.edu

ABSTRACT

We present the first systematic analysis of the characteristics of patch search spaces for automatic patch generation systems. We analyze the search spaces of two current state-of-the-art systems, SPR and Prophet, with 16 different search space configurations. Our results are derived from an analysis of 1104 different search spaces and 768 patch generation executions. Together these experiments consumed over 9000 hours of CPU time on Amazon EC2.

The analysis shows that 1) correct patches are sparse in the search spaces (typically at most one correct patch per search space per defect), 2) incorrect patches that nevertheless pass all of the test cases in the validation test suite are typically orders of magnitude more abundant, and 3) leveraging information other than the test suite is therefore critical for enabling the system to successfully isolate correct patches.

We also characterize a key tradeoff in the structure of the search spaces. Larger and richer search spaces that contain correct patches for more defects can actually cause systems to find fewer, not more, correct patches. We identify two reasons for this phenomenon: 1) increased validation times because of the presence of more candidate patches and 2) more incorrect patches that pass the test suite and block the discovery of correct patches. These fundamental properties, which are all characterized for the first time in this paper, help explain why past systems often fail to generate correct patches and help identify challenges, opportunities, and productive future directions for the field.

Categories and Subject Descriptors

D.2.5 [**SOFTWARE ENGINEERING**]: Testing and Debugging

Keywords

Program repair, Patch generation, Search space

1. INTRODUCTION

Software defects are a prominent problem in software development efforts. Motivated by the prospect of reducing human developer involvement, researchers have developed a range of techniques that are designed to automatically correct defects. In this paper we focus on generate and validate patch generation systems, which work with a test suite of test cases, generate a set of candidate patches, then test the patched programs against the test suite to find a patch that validates [27, 16, 38, 28, 14, 29, 17, 18].

Patch quality is a key issue for generate and validate systems. Because the patches are validated only against the test cases in the test suite, there is no guarantee that the patch will enable the program to produce correct results for other test cases. Indeed, recent research has shown that 1) the majority of patches accepted by many current generate and validate systems fail to generalize to produce correct results for test cases outside the validation test suite [29, 7, 35] and 2) accepted patches can have significant negative effects such as the introduction of new integer and buffer overflow security vulnerabilities, undefined accesses, memory leaks, and the elimination of core application functionality [29]. These negative effects highlight the importance of generating not just *plausible patches* (we define plausible patches to be patches that pass all of the test cases in the patch validation test suite) but *correct patches* that do not have latent defects and do not introduce new defects or vulnerabilities.

A rich search space that contains correct patches for target defects can be critical to the success of any automatic patch generation system. Indeed, recent research indicates that impoverished search spaces that contain very few correct patches is one of the reasons for the poor performance of some prominent previous patch generation systems [16, 38, 29]. While more recent systems work with patch spaces that contain significantly more correct patches [17, 18], continued progress in the field requires even richer patch spaces that contain more successful correct patches. But these richer spaces may also complicate the ability of the system to identify correct patches within the larger sets of plausible but incorrect patches.

1.1 SPR and Prophet

SPR [17] and Prophet [18] are two current state-of-the-art generate and validate patch generation systems. Both systems apply transformations to statements identified by an error localization algorithm to obtain patches that they validate against a test suite. SPR uses a set of hand-coded patch prioritization heuristics. Prophet uses machine learning to characterize features of previously successful human patches and prioritizes candidate patches according to these features. The goal is to prioritize a correct patch as the first patch to validate.

The baseline SPR and Prophet search spaces contain correct patches for 19 out of 69 defects in a benchmark set of defects from eight open source projects [17, 18].[1] For 11

[1]The paper that presented this benchmark set states that the benchmark set contains 105 defects. An examination of the relevant commit logs and applications indicates that 36

of these defects, the first SPR patch to validate is a correct patch. For 15 of these defects, the first Prophet patch to validate is a correct patch. The GenProg [16], AE [38], RSRepair [28], and Kali [29] systems, in contrast, produce correct patches for only 1 (GenProg, RSRepair) or 2 (AE, Kali) of the defects in this benchmark suite [29]. Moreover, the correct SPR and Prophet patches for the remaining defects lie outside the GenProg, RSRepair, and AE patch spaces, which suggests that these systems will *never* be able to produce correct patches for these remaining defects. We note that this benchmark set was developed not by us, but by others in an attempt to obtain a large, unbiased, and realistic benchmark set [16].

As these results highlight, the characteristics of the patch search space are central to the success of the patch generation system. But despite the importance of the search space in the overall success of the patch generation system, we have been able to find no previous systematic investigation of how the structure of the patch search space influences critical characteristics such as the density of correct and plausible patches and the ability of the system to identify correct patches within the broader class of plausible (but potentially incorrect) patches. Given the demonstrated negative effects of plausible but incorrect patches generated by previous systems [29], these characteristics play a critical role in the overall success or failure of any patch generation system that may generate plausible patches with such negative effects.

1.2 Patch Search Space Analysis

We present a systematic analysis of the SPR and Prophet patch search spaces. This analysis focuses on the density of correct and plausible patches in the search spaces, on the ability of SPR and Prophet to prioritize correct patches, and on the consequences of two kinds of changes, both of which increase the size of the search space: 1) increases in the number of candidate program statements to which patches are applied and 2) new program transformation operators that can generate additional patches. Starting from the SPR and Prophet baseline search spaces, these changes make it possible to construct a collection of search spaces, with each search space characterized by a combination of the set of transformations and the number of candidate program statements that together generate the search space.

We perform our analysis on a benchmark set of 69 real world defects in eight large open source applications. This benchmark set was used to evaluate many previous patch generation systems [16, 38, 29, 28, 17, 18]. For each defect, we first analyze the full search space to determine whether or not it contains a correct patch. We acknowledge that, in general, determining whether a specific patch corrects a specific defect can be difficult (or in some cases not even well defined). But for the defects in the benchmark set, this never happens — the correct patches that are within the search spaces are all small, all match the corresponding developer patches, and the distinction between correct and incorrect patches is clear.

Because the full search spaces can, in general, be too large to exhaustively search within any reasonable time limit, we also consider the subset of the search spaces that can be explored by SPR and Prophet within a reasonable timeout, in this paper 12 hours. Working with these explored subsets of spaces, we analyze the number of plausible and correct patches that each subset contains and the effect of the SPR and Prophet patch prioritization on the ability of these systems to identify correct patches within the much larger sets of plausible but incorrect patches in these search spaces.

of these defects are actually deliberate functionality changes, not defects. In this paper we focus on the remaining 69 actual defects as within the scope of the paper.

1.3 Results Overview

Our experimental results indicate that:

- **Sparse Correct Patches:** Correct patches occur only sparsely within the search spaces. For 45 of the 69 defects, the search spaces contain no correct patches. For 15 of the remaining 24 defects, the search spaces contain at most 1 correct patch. The largest number of correct patches for any defect in any search space is 4.

- **Relatively Abundant Plausible Patches:** In comparison with correct patches, plausible (but incorrect) patches are relatively abundant. For all of the benchmarks except php, the explored search spaces typically contain hundreds up to a thousand times more plausible patches than correct patches. These numbers highlight the difficulty of isolating correct patches among the large sets of plausible but incorrect patches.

  The explored search spaces for php, in contrast, typically contain only tens of times more plausible patches than correct patches. And for three of the php defects, all of the (one or two) plausible patches are correct. The density of plausible patches is obviously related to the strength of the validation test suite — weak test suites filter fewer incorrect patches. We attribute the difference in plausible patch density between php and the other benchmarks to the strength of the php test suite — the php test suite contains an order of magnitude more test cases than any other benchmark.

- **SPR and Prophet Effectiveness:** The SPR and Prophet patch prioritization mechanisms are both effective at isolating correct patches within the explored plausible patches. Despite the relatively scarcity of correct patches, with the baseline search space, correct patches for 14 defects are within the first ten patches to validate for SPR; correct patches for 16 defects are within the first ten patches to validate for Prophet.

- **Search Space Tradeoffs:** Increasing the search space beyond the SPR and Prophet baseline increases the number of defects that have a correct patch in the search space. But it does not increase the ability of SPR and Prophet to find correct patches for more defects — in fact, these increases often cause SPR and Prophet to find correct patches for *fewer* defects!

  We attribute this phenomenon to the following tradeoff. Increasing the search space also increases the number of candidate patches and may increase the number of plausible patches. The increased number of candidate patches consumes patch evaluation time and reduces the density of the correct patches in the search space. The increased number of plausible but incorrect patches increases the chance that such patches will block the correct patch (i.e., that the system will encounter plausible but incorrect patches as the first patches to validate).

These facts highlight the importance of including information other than the test suite in the patch evaluation process. SPR includes information in the form of hand-coded patch prioritization heuristics. Prophet leverages information available via machine learning from successful human patches. This information is responsible for the ability of these systems to successfully identify correct patches in the baseline search space. The results also highlight that there is still room for improvement, especially with richer search spaces that contain correct patches for more defects.

Previous Systems: These facts also help explain past results from other systems. GenProg, AE, and RSRepair generate very few correct patches [29]. Part of the explanation is that the search space exploration algorithms for these systems are no better than random search [28].[2] Once one appreciates the relative abundance of plausible but incorrect patches and the relative scarcity of correct patches, it is clear that any algorithm that is no better than random has very little chance of consistently delivering correct patches without very strong test suites. And indeed, the majority of the patches from these previous systems simply delete functionality and do not actually fix the defect [29].

ClearView: ClearView, in sharp contrast, does leverage information other than the test suite, specifically learned invariants from previous successful executions [27]. The results indicate a dramatic improvement in ClearView's ability to locate patches that eliminate security vulnerability defects. For nine of the ten defects on which ClearView was evaluated, ClearView successfully patched the defect after evaluating at most three candidate patches. These results highlight how targeting a defect class and leveraging fruitful sources of information can dramatically increase the successful patch density.

1.4 Future Directions

Our results highlight the scalability challenges associated with generalizing existing search spaces to include correct patches for more defects. One obvious future direction, deployed successfully in past systems [30, 19, 6], is to address scalability issues by developing smaller, more precisely targeted search spaces for specific classes of defects. An alternative is to infer transformation operations from correct human patches (instead using manually defined transformations). The goal is to obtain a search space that contains correct patches for common classes of commonly occurring defects while still remaining tractable.

More broadly, it is now clear that generate and validate systems must exploit information beyond current validation test suites if they are to successfully correct any but the most trivial classes of defects [27, 18, 29]. One prominent direction is to exploit existing correct code in large code repositories to obtain new correct patches, either via sophisticated machine learning techniques that learn to recognize or even automatically generate correct code, automatically transferring correct code (either within or between applications), or even generalizing and combining multiple blocks of correct code throughout the entire software ecosystem. Encouraging initial progress has been made in all of these directions [18, 34, 36]. The current challenge is to obtain more sophisticated patches for broader classes of defects.

An orthogonal direction is to obtain stronger test suites or even explicit specifications that can more effectively filter incorrect patches. One potential approach is to observe correct input/output pairs to learn to recognize or even automatically generate correct outputs for (potentially narrowly targeted) classes of inputs. Another approach leverages the availability of multiple implementations of the same basic functionality (for example, multiple image rendering applications) to recognize correct outputs. Combining either of these two capabilities with automatic input generation could enable the automatic generation of much stronger test suites (with potential applications far beyond automatic patch generation). Specification mining may also deliver (potentially partial) specifications that can help filter incorrect patches.

1.5 Contributions

This paper makes the following contributions:

- **Patch Space Analysis:** It presents an analysis of the patch search spaces of SPR and Prophet, including how these patch spaces respond to the introduction of new transformation operators and increases in candidate program statements. The analysis characterizes:
  - **Correct Patches:** The density at which correct patches occur in the full search spaces and the explored space subsets.
  - **Plausible Patches:** The density at which plausible patches occur in the explored space subsets.
  - **Patch Prioritization:** The effectiveness of the SPR and Prophet patch prioritization mechanisms at isolating the few correct patches within the much larger set of plausible patches (the vast majority of which are not correct).

  This paper presents the first characterization of how correct and plausible patch densities respond to increases in the size and sophistication of the search space. It also presents the first characterization of how these search space changes affect the ability of the patch generation system to identify the few correct patches within the much larger sets of plausible but incorrect patches.

- **Tradeoff:** It identifies and presents results that characterize a tradeoff between the size and sophistication of the search space and the ability of the patch generation system to identify correct patches. To the best of our knowledge, this is the first characterization of this tradeoff.

- **Results:** It presents experimental results from SPR and Prophet with different search spaces. These results are derived from an analysis of 1104 different search spaces for the 69 benchmark defects (we consider 16 search spaces for each defect) and 768 patch generation executions for the 24 defects whose correct patches are inside any of the considered search spaces (we run SPR and Prophet for each of the 16 search spaces for each of these 24 defects). Together, these executions consumed over 9000 hours of CPU time on Amazon EC2 cluster. The results show:

[2]Other parts of the explanation include search spaces that apparently contain very few correct patches and errors in the patch infrastructure that cause the system to accept patches that do not even pass the test cases in the test suites used to validate the patches [29].

– **Sparse Correct Patches:** Correct patches occur very sparsely within the patch spaces, with typically no more than one correct patch in the search space for a given defect.

– **Relatively Abundant Plausible Patches:** Depending on the strength of the validation test suite, plausible patches are either two to three orders of magnitude or one order of magnitude more abundant than correct patches.

– **Patch Prioritization Effectiveness:** The SPR and Prophet patch prioritization algorithms exhibit substantial effectiveness at isolating correct patches within the large set of plausible patches (most of which are incorrect).

– **Challenges of Rich Search Spaces:** The challenges associated with successfully searching such spaces include increased testing overhead and increased chance of encountering plausible but incorrect patches that block the subsequent discovery of correct patches.

Progress in automatic patch generation systems requires the development of new, larger, and richer patch search spaces that contain correct patches for larger classes of defects. This paper characterizes, for the first time, how current state-of-the-art patch generation systems respond to changes in the size and sophistication of their search spaces. It therefore identifies future productive directions for the field and provides a preview of the challenges that the field will have to overcome to develop systems that work productively with more sophisticated search spaces to successfully patch broader classes of defects.

2. SPR AND PROPHET

We next present an overview of the two automatic patch generation systems, SPR [17] and Prophet [18], whose search spaces and search algorithms we analyze.

2.1 Design Overview

SPR and Prophet start with a defective program to patch and a validation test suite. The test suite contains 1) a set of negative test cases which the original program does not pass (these test cases expose the defect in the program) and 2) a set of positive test cases which the original program already passes (these test cases prevent regression). The test cases include correct outputs for every input (the negative test cases produce different incorrect outputs). The system generates patches for the program with the following steps:
Error Localization: The system first uses an error localizer to identify a set of candidate program statements to modify. The error localizer recompiles the given application with additional instrumentation. It inserts a call back before each statement in the source code to record a positive counter value as the timestamp of the statement execution. The error localizer then invokes the recompiled application on all test cases and produces, based on the recorded timestamp values, a prioritized list of target statements to modify. The error localizer prioritizes statements that are 1) executed with more negative test cases, 2) executed with fewer positive test cases, and 3) executed later during executions with negative test cases. See the SPR and Prophet papers for more details [17, 18].

Apply Transformations: The system then applies a set of transformations to the identified program statements to generate the search space of candidate patches. SPR and Prophet consider the following transformation schemas [17]:

- **Condition Refinement:** Given a target if statement to patch, the system transforms the condition of the if statement by conjoining or disjoining an additional condition to the original if condition. The following two patterns implement the transformation:

  ```
  if (C) { ... } => if (C && P) { ... }
  if (C) { ... } => if (C || P) { ... }
  ```

  Here `if (C) { ... }` is the target statement to patch in the original program. `C` is the original condition that appears in the program. `P` is a new condition produced by a condition synthesis algorithm [17, 18].

- **Condition Introduction:** Given a target statement, the system transforms the program so that the statement executes only if a guard condition is true. The following pattern implements the transformation:

  ```
  S => if (P) S
  ```

  Here `S` is the target statement to patch in the original program and `P` is a new synthesized condition.

- **Conditional Control Flow Introduction:** Before a target statement, the system inserts a new control flow statement (return, break, or goto an existing label) that executes only if a guard condition is true. The following patterns implement the transformation:

  ```
  S => if (P) break; S
  S => if (P) continue; S
  S => if (P) goto L; S
  ```

  Here `S` is the target statement to patch in the original program, `P` is a new synthesized condition, and `L` is an existing label in the procedure containing `S`.

- **Insert Initialization:** Before a target statement, the system inserts a memory initialization statement.

- **Value Replacement:** Given a target statement, replace an expression in the statement with another expression.

- **Copy and Replace:** Given a target statement, the system copies an existing statement to the program point before the target statement and then applies a Value Replacement transformation to the copied statement.

Condition Synthesis: The baseline versions of SPR and Prophet work with synthesized conditions `P` of the form `E == K` and `E != K`. Here `E` is a *check expression*, which we define as either a local variable, a global variable, or a sequence of structure field accesses. Each check expression `E` must appear in the basic block containing the synthesized condition. `K` is a *check constant*, which we define as a constant drawn from the set of values that the check expression `E` takes on during the instrumented executions of the unpatched program on the negative test cases.

Value Replacement: The baseline versions of SPR and Prophet replace either 1) one variable in the target statement with another variable that appears in the basic block containing the statement, 2) an invoked function in the statement with another function that 1) has a compatible type signature and 2) is invoked or declared in the source code file containing the statement (or a header file included in this source code file), or 3) a constant in the statement with another constant that appears in the function containing the statement.

Evaluate Candidate Patches: The system then evaluates candidate patches in the search space against the supplied test cases. To efficiently explore the search space, SPR and Prophet use *staged program repair* [17, 18]. At the first stage, the system operates with parameterized candidate patch templates, which may contain an abstract expression. It instantiates and evaluates concrete patches from a template only if the system determines that there may be a concrete patch from the template that passes the test cases.

For the first three transformation schemas (these schemas manipulate conditions), the system first introduces an abstract condition into the program and determines whether there is a sequence of branch directions for the abstract condition that will enable the patched program to pass the test cases. If so, the system then synthesizes concrete conditions to generate patches.

2.2 Extensions

We implement three extensions to the SPR and Prophet search spaces: considering more candidate program statements to patch, synthesizing more sophisticated conditions, and evaluating more complicated value replacement transformations.

More Program Statements to Patch: The baseline SPR and Prophet configurations consider the first 200 program statements identified by the error localizer. We modify SPR and Prophet to consider the first 100, 200, 300, and 2000 statements.

Condition Synthesis Extension (CExt): We extend the baseline SPR and Prophet condition synthesis algorithm to include the "`<`" and "`>`" operators and to also consider comparisons between two check expressions (e.g., `E < K`, $\texttt{E}_1$ `==` $\texttt{E}_2$, and $\texttt{E}_1$ `>` $\texttt{E}_2$, where `E`, $\texttt{E}_1$, and $\texttt{E}_2$ are check expressions and `K` is a check constant). In the rest of this paper, we use "CExt" to denote this search space extension.

Value Replacement Extension (RExt): We extend the baseline SPR and Prophet replacement transformations to also replace a variable or a constant in the target statement with an expression that is composed of either 1) a unary operator and an atomic value (i.e., a variable or a constant) which appears in the basic block containing the statement or 2) a binary operator and two such atomic values. The operators that SPR and Prophet consider are "`+`", "`-`", "`*`", "`==`", "`!=`", and "`&`". In the rest of this paper, we use "RExt" to denote this search space extension.

2.3 SPR Prioritization Order

SPR uses a set of hand-coded heuristics to prioritize its search of the generated patch space. These heuristics prioritize patches in the following order: 1) patches that change only a branch condition (e.g., tighten and loosen a condition), 2) patches that insert an if-statement before the first statement of a compound statement (i.e., C code block), 3) patches that insert an if-guard around a statement, 4) patches that replace a statement, insert an initialization statement, insert an if-statement, or insert a statement before the first statement of a compound statement, and 5) finally all the remaining patches. For each kind of patch, it prioritizes statements to patch in the error localization order.

2.4 Prophet Prioritization Order

Prophet searches the same patch space as SPR, but works with a corpus of correct patches from human developers. It processes this corpus to learn a probabilistic model that assigns a probability to each candidate patch in the search space. This probability indicates the likelihood that the patch is correct. It then uses this model to prioritize its search of the patch space.

A key idea behind Prophet is that patch correctness depends on not just the patch itself, but also on how the patch interacts with the surrounding code:

- **Extract Features:** For each patch in the corpus, Prophet analyzes a structural diff of the abstract syntax trees of the original and patched code to extract both 1) features which summarize how the patch modifies the program given characteristics of the surrounding code and 2) features which summarize relationships between roles that values accessed by the patch play in the original unpatched program and in the patch.

- **Learn Model Parameters:** Prophet operates with a parameterized log-linear probabilistic model in which the model parameters can be interpreted as weights that capture the importance of different features. Prophet learns the model parameters via maximum likelihood estimation, i.e., the Prophet learning algorithm attempts to find parameter values that maximize the probability of observing the collected training database in the probabilistic model.

Prophet uses the trained model to rank the patches according to its learned model of patch correctness, then evaluates the patches in that order. Previous results (as well as additional results presented in this paper) show that this learned patch correctness model outperforms SPR's heuristics [18]. This result highlights how leveraging information available in existing large software development projects can significantly improve our ability to automatically manipulate large software systems.

3. STUDY METHODOLOGY

We next present our study methodology.

3.1 Benchmark Applications

We use a benchmark set of 69 real world defects to perform our search space study. Those defects are from eight large open source applications, libtiff, lighttpd, the php interpreter, gmp, gzip, python, wireshark, and fbc [16]. Note that the original benchmark set also includes 36 ostensible defects which correspond to deliberate functionality changes, not defects, during the application development [17]. We exclude those functionality changes as outside the scope of our study because they are not actual defects.

Table 1 summarizes our benchmark defects. The first column (App.) presents the name of each application. The second column (LoC) presents the number of lines of code in the

App.	LoC	Tests	Defects	SPR	Prophet
libtiff	77k	78	8	1/3	2/3
lighttpd	62k	295	7	0/0	0/0
php	1046k	8471	31	9/13	10/13
gmp	145k	146	2	1/1	1/1
gzip	491k	12	4	0/1	1/1
python	407k	35	9	0/0	0/0
wireshark	2814k	63	6	0/0	0/0
fbc	97k	773	2	0/1	1/1
Total			69	11/19	15/19

Table 1: Benchmark Applications

Defect	Localization Rank	RExt	CExt
lighttpd-2661-2662	1926	No	No
lighttpd-1913-1914	280	No	Yes
python-70056-70059	214	No	Yes
python-69934-69935	136	Yes	No
gmp-14166-14167	226	Yes	No

Table 2: Search Space Extensions

application. The third column (Tests) presents the number of the test cases in the supplied test suite of the application. php is the outlier, with an order of magnitude more test cases than any other application. The fourth column (Defects) presents the number of defects in the benchmark set for each application.

The fifth column (SPR) and the sixth column (Prophet) present the patch generation results for the baseline versions of SPR [17] and Prophet [18], respectively. Each entry is of the form "X/Y", where Y is the number of defects whose correct patches are inside the search space, while X is the number of defects for which the system automatically generates a correct patch as the first generated patch.

3.2 Experimental Setup

Configure Systems: We run SPR and Prophet on each of the 16 different search space configurations derived from all possible combinations of 1) working with the first 100, 200, 300, or 2000 program statements identified by the error localizer, 2) whether to enable value replacement extension (RExt), and 3) whether to enable condition synthesis extension (CExt).

We run all of our experiments except those of fbc on Amazon EC2 Intel Xeon 2.6GHz machines running Ubuntu-64bit server 14.04. fbc runs only in 32-bit environments, so we run all fbc experiments on EC2 Intel Xeon 2.4GHz machines running Ubuntu-32bit 14.04.

Generate Search Spaces: For each search space and each of the 69 defects in the benchmark set, we run the configured SPR and Prophet to generate and print the search space for that defect. We then analyze the generated search space and determine whether the space contains a correct patch for the defect.

With all three search space extensions, the generated SPR and Prophet search spaces contain correct patches for five more defects (i.e., 24 defects in total) than the baseline search space. Table 2 summarizes these five defects. The first column (Defect) contains entries of the form X-Y-Z, where X is the name of the application that contains the defect, Y is the defective revision in the application repository, and Z is the reference fixed revision in the repository. The second column (Localization Rank) presents the error localization rank of the modified program statement in the correct patch for the defect. The third column (RExt) presents whether the correct patches for the defect require the RExt extension (value replacement extension) The fourth column (CExt) presents whether the correct patches for the defect require the CExt extension (condition synthesis extension).

Generate Patches: For each search space and each of the 24 defects with correct patches in the search space, we run SPR and Prophet to explore the search space for the defect. For each run, we record all of the plausible patches that the system discovers within the 12 hour timeout.

Analyze Patches: For each defect, we analyze the generated plausible patches for the defect to determine whether the patch is correct or incorrect.

4. EXPERIMENTAL RESULTS

We next present the experimental results. php is an outlier with a test suite that contains an order of magnitude more test cases than the other applications. We therefore separate the php results from the results from other benchmarks. We present the result summary for all of the 24 defects for which any of the search spaces contains a correct patch. See Appendix for the detailed results of each defect in all different search space configurations.

Table 3 presents a summary of the results for all of the benchmarks except php. Table 4 presents a summary of the results for the php benchmark. Each row presents patch generation results for SPR or Prophet with one search space configuration. The first column (System) presents the evaluated system (SPR or Prophet). The second column (Loc. Limit) presents the number of considered candidate program statements to patch under the configuration. The third column (Space Extension) presents the transformation extensions that are enabled in the configuration: No (no extensions, baseline search space), CExt (condition synthesis extension), RExt (value replacement extension), or RExt+CExt (both).

The fourth column (Correct In Space) presents the number of defects with correct patches that lie inside the full search space for the corresponding configuration. The fifth column (Correct First) presents the number of defects for which the system finds the correct patch as the *first* patch that validates against the test suite.

Each entry of the sixth column (Plausible & Blocked) is of the form X(Y). Here X is the number of defects for which the system discovers a plausible but incorrect patch as the first patch that validates. Y is the number of defects for which a plausible but incorrect patch blocked a subsequent correct patch (i.e., Y is the number of defects for which 1) the system discovers a plausible but incorrect patch as the first patch that validates and 2) the full search space contains a correct patch for that defect).

Each entry of the seventh column (Timeout) is also of the form X(Y). Here X is the number of defects for which the system does not discover any plausible patch within the 12 hour timeout. Y is the number of defects for which 1) the system does not discover a plausible patch and 2) the full search space contains a correct patch for that defect.

The eighth column (Space Size) presents the average number of candidate patch templates in the search space over all of the 24 considered defects. Note that SPR and Prophet

System	Loc. Limit	Space Extension	Correct		Plausible & Blocked	Timeout	Space Size	Correct Rank	Plausible in 12h	Correct in 12h
			In Space	First						
SPR	100	No	4	1	7(3)	3(0)	20068.5	4614.0	8(2747)	4(5)
SPR	100	CExt	4	1	7(3)	3(0)	20068.5	4614.0	8(11438)	3(4)
SPR	100	RExt	4	1	7(3)	3(0)	21999.8	6004.8	8(2742)	4(5)
SPR	100	RExt+CExt	4	1	7(3)	3(0)	21999.8	6004.8	8(11192)	3(4)
SPR	200	No	6	2	7(4)	2(0)	46377.6	17889.5	9(2558)	6(8)
SPR	200	CExt	6	2	7(4)	2(0)	46377.6	17889.5	9(10823)	4(6)
SPR	200	RExt	7	2	7(4)	2(1)	52864.3	24759.9	9(3753)	6(8)
SPR	200	RExt+CExt	7	2	7(4)	2(1)	52864.3	24759.9	9(10855)	4(6)
SPR	300	No	6	1	8(5)	2(0)	73559.6	22960.0	9(2818)	6(8)
SPR	300	CExt	8	1	8(6)	2(1)	73559.6	30761.8	9(10237)	4(6)
SPR	300	RExt	8	1	8(6)	2(1)	82187.2	32951.4	9(2069)	7(8)
SPR	300	RExt+CExt	10	1	8(7)	2(2)	82187.2	37427.4	9(10455)	5(6)
SPR	2000	No	7	2	7(5)	2(0)	523753.8	157038.4	9(751)	5(6)
SPR	2000	CExt	9	2	7(6)	2(1)	523753.8	156495.1	9(6123)	4(5)
SPR	2000	RExt	9	2	7(6)	2(1)	574325.1	200996.7	9(657)	5(6)
SPR	2000	RExt+CExt	11	2	7(7)	2(2)	574325.1	192034.0	9(5831)	4(5)
Prophet	100	No	4	4	4(0)	3(0)	20068.5	589.2	8(2481)	4(5)
Prophet	100	CExt	4	3	5(1)	3(0)	20068.5	589.2	8(11901)	3(4)
Prophet	100	RExt	4	4	4(0)	3(0)	21999.8	520.5	8(2183)	4(5)
Prophet	100	RExt+CExt	4	3	5(1)	3(0)	21999.8	520.5	8(11595)	3(4)
Prophet	200	No	6	5	4(1)	2(0)	46377.6	11382.8	9(2564)	5(7)
Prophet	200	CExt	6	4	5(2)	2(0)	46377.6	11382.8	9(10968)	4(6)
Prophet	200	RExt	7	5	4(1)	2(1)	52864.3	19581.0	9(1939)	5(6)
Prophet	200	RExt+CExt	7	4	5(2)	2(1)	52864.3	19581.0	9(10928)	4(5)
Prophet	300	No	6	4	5(2)	2(0)	73559.6	11997.2	9(2555)	5(7)
Prophet	300	CExt	8	3	6(4)	2(1)	73559.6	14466.8	9(10948)	4(6)
Prophet	300	RExt	8	4	5(3)	2(1)	82187.2	25769.1	9(1548)	5(6)
Prophet	300	RExt+CExt	10	3	6(5)	2(2)	82187.2	25455.1	9(10886)	4(5)
Prophet	2000	No	7	4	5(3)	2(0)	523753.8	188588.4	9(1229)	5(7)
Prophet	2000	CExt	9	3	6(5)	2(1)	523753.8	156555.8	9(8208)	4(6)
Prophet	2000	RExt	9	3	6(5)	2(1)	574325.1	170715.4	9(1216)	5(6)
Prophet	2000	RExt+CExt	11	2	7(7)	2(2)	574325.1	148288.4	9(7919)	4(5)

Table 3: Patch Generation Results with Search Space Extensions (excluding php)

may instantiate multiple concrete patches with the staged program repair technique from a patch template that contains an abstract expression (See Section 2.1). This column shows how the size of the search space grows as a function of the number of candidate statements to patch and the two extensions. Note that the CExt transformation extension does not increase the number of patch templates. Instead it increases the number of concrete patches which each patch template generates. The ninth column (Correct Rank) presents the average rank of the first patch template that generates a correct patch in the search space over all of those defects for which at least one correct patch is inside the search space. Note that the correct rank increases as the size of the search space increases.

Each entry of the tenth column (Plausible in 12h) is of the form X(Y). Here X is the number of defects for which the system discovers a plausible patch within the 12 hour timeout. Y is the sum, over the all of the 24 considered defects, of the number of plausible patches that the system discovers within the 12 hour timeout.

Each entry of the eleventh column (Correct in 12h) is of the form X(Y). Here X is the number of defects for which the system discovers a correct patch (blocked or not) within the 12 hour timeout. Y is the number of correct patches that the system discovers within the 12 hour timeout.

4.1 Plausible and Correct Patch Density

An examination of the tenth column (Plausible in 12h) in Tables 3 and 4 highlights the overall plausible patch densities in the search spaces. For the benchmarks without php, the explored search spaces typically contain hundreds up to a thousand plausible patches per defect. For php, in contrast, the explored search spaces typically contain tens of plausible patches per defect. We attribute this significant difference in the plausible patch density to the quality of the php test suite and its resulting ability to successfully filter out otherwise plausible but incorrect patches. Indeed, for three php defects, the php test suite is strong enough to filter out all of the patches in the explored search spaces except the correct patch.

An examination of the eleventh column (Correct in 12h) in Tables 3 and 4 highlights the overall correct patch densities in the explored search spaces. In sharp contrast to the plausible patch densities, the explored search spaces contain, on average, less than two correct patches per defect for all of the benchmarks including php. There are five defects with as many as two correct patches in any search space and one defect with as many as four correct patches in any search space. The remaining defects contain either zero or one correct patch across all of the search spaces.

4.2 Search Space Tradeoffs

An examination of the fourth column (Correct In Space) in Table 3 shows that the number of correct patches in the full search space increases as the size of the search space increases (across all benchmarks except php). But an examination of the fifth column (Correct First) indicates that that this increase does not translate into an increase in the

System	Loc. Limit	Space Extension	Correct		Plausible & Blocked	Timeout	Space Size	Correct Rank	Plausible in 12h	Correct in 12h
			In Space	First						
SPR	100	No	12	8	3(3)	2(1)	13446.5	4157.8	11(237)	9(14)
SPR	100	CExt	12	8	4(4)	1(0)	13446.5	4157.8	12(415)	10(12)
SPR	100	RExt	12	8	4(4)	1(0)	14026.7	4360.8	12(288)	11(15)
SPR	100	RExt+CExt	12	8	4(4)	1(0)	14026.7	4360.8	12(421)	10(12)
SPR	200	No	13	9	3(3)	1(1)	26512.0	7369.8	12(197)	10(15)
SPR	200	CExt	13	9	3(3)	1(1)	26512.0	7369.8	12(330)	10(13)
SPR	200	RExt	13	9	3(3)	1(1)	28158.2	7984.5	12(200)	10(15)
SPR	200	RExt+CExt	13	9	3(3)	1(1)	28158.2	7984.5	12(323)	10(13)
SPR	300	No	13	8	4(4)	1(1)	41859.8	10915.2	12(176)	9(14)
SPR	300	CExt	13	8	4(4)	1(1)	41859.8	10915.2	12(305)	9(12)
SPR	300	RExt	13	8	4(4)	1(1)	44631.1	12440.9	12(179)	9(14)
SPR	300	RExt+CExt	13	8	4(4)	1(1)	44631.1	12440.9	12(313)	9(12)
SPR	2000	No	13	5	2(2)	6(6)	327905.6	81570.5	7(58)	5(6)
SPR	2000	CExt	13	5	2(2)	6(6)	327905.6	81570.5	7(126)	5(6)
SPR	2000	RExt	13	5	3(3)	5(5)	356104.8	83997.9	8(59)	5(6)
SPR	2000	RExt+CExt	13	5	2(2)	6(6)	356104.8	83997.9	7(127)	5(6)
Prophet	100	No	12	8	3(3)	2(1)	13446.5	2599.4	11(279)	11(15)
Prophet	100	CExt	12	6	6(6)	1(0)	13446.5	2599.4	12(466)	11(11)
Prophet	100	RExt	12	9	3(3)	1(0)	14026.7	3433.8	12(327)	11(15)
Prophet	100	RExt+CExt	12	6	6(6)	1(0)	14026.7	3433.8	12(458)	11(11)
Prophet	200	No	13	10	3(3)	0(0)	26512.0	3522.1	13(285)	13(18)
Prophet	200	CExt	13	7	6(6)	0(0)	26512.0	3522.1	13(447)	12(13)
Prophet	200	RExt	13	10	3(3)	0(0)	28158.2	4504.4	13(296)	12(17)
Prophet	200	RExt+CExt	13	7	6(6)	0(0)	28158.2	4504.4	13(434)	12(13)
Prophet	300	No	13	10	3(3)	0(0)	41859.8	4319.6	13(280)	13(18)
Prophet	300	CExt	13	7	6(6)	0(0)	41859.8	4319.6	13(425)	12(13)
Prophet	300	RExt	13	10	3(3)	0(0)	44631.1	5403.1	13(283)	12(17)
Prophet	300	RExt+CExt	13	7	6(6)	0(0)	44631.1	5403.1	13(422)	12(13)
Prophet	2000	No	13	7	2(2)	4(4)	327905.6	21118.6	9(117)	7(10)
Prophet	2000	CExt	13	4	4(4)	5(5)	327905.6	21118.6	8(153)	6(6)
Prophet	2000	RExt	13	6	2(2)	5(5)	356104.8	25168.5	8(104)	6(9)
Prophet	2000	RExt+CExt	13	4	4(4)	5(5)	356104.8	25168.5	8(183)	6(6)

Table 4: Patch Generation Results with Search Space Extensions (php only)

ability of SPR or Prophet to actually find these correct patches as the first patch to validate. In fact, the ability of SPR and Prophet to isolate a correct patch as the first patch to validate reaches a maximum at 200 candidate statements with no extensions, then (in general) decreases from there as the size of the search space increases. For php, Table 4 shows that the number of correct patches in the space does not significantly increase with the size of the search space, but that the drop in the number of correct patches found as the first patch to validate is even more significant. Indeed, the 200+No Prophet configuration finds 10 correct patches as the first patch to validate, while the largest 2000+RExt+CExt configuration finds only four!

We attribute these facts to an inherent tradeoff in the search spaces. Expanding the search spaces to include more correct patches also includes more implausible and plausible but incorrect patches. The implausible patches consume validation time (extending the time required to find the correct patches), while the plausible but incorrect patches block the correct patches. This trend is visible in the Y entries in the sixth column in Table 3 (Plausible & Blocked) (these entries count the number of blocked correct patches), which generally increase as the size of the search space increases.

Tables 3 and 4 show how this tradeoff makes the baseline SPR and Prophet configurations perform best despite working with search spaces that contain fewer correct patches. Increasing the candidate statements beyond 200 never increases the number of correct patches that are first to validate. Applying the CExt and RExt extensions also never increases the number of correct patches that are first to validate.

Our results highlight two challenges that SPR and Prophet (and other generate and validate systems) face when generating correct patches:

- **Weak Test Suites:** The test suite provides incomplete coverage. The most obvious problem of the weak test suite is that it may accept incorrect patches. Our results show that (especially for larger search spaces) plausible but incorrect patches often block correct patches. For example, when we run Prophet with the baseline search space (200+No), there are only 4 defects whose correct patches are blocked; when we run Prophet with the largest search space (2000+RExt+CExt), there are 11 defects whose correct patches are blocked.
  A more subtle problem is that weak test suites may increase the validation cost of plausible but incorrect patches. For such a patch, SPR or Prophet has to run the patched application on all test cases in the test suite. If a stronger test suite is used, SPR and Prophet may invalidate the patch with one test case and skip the remaining test cases.
- **Search Space Explosion:** A large search space contains many candidate patch templates and our results show that it may be intractable to validate all of the candidates. For example, with the baseline search space (200+No), Prophet times out for only two defects (whose correct patches are outside the

Search Space	SPR	Prophet	Random (SPR)	Random (Prophet)
100+No	52 / 3	38 / 4	65.0 / 1.4	65.0 / 1.4
100+CExt	65 / 2	53 / 3	73.0 / 1.0	73.0 / 1.0
100+RExt	52 / 3	38 / 4	65.1 / 1.4	65.1 / 1.4
100+RExt+CExt	65 / 2	53 / 3	73.0 / 1.0	73.0 / 1.0
200+No	59 / 4	45 / 5	73.5 / 2.7	76.2 / 2.1
200+CExt	66 / 3	54 / 4	78.1 / 1.7	78.1 / 1.7
200+RExt	59 / 4	45 / 5	76.2 / 2.1	75.9 / 2.1
200+RExt+CExt	66 / 3	54 / 4	77.8 / 1.7	77.8 / 1.7
300+No	60 / 5	50 / 5	80.2 / 2.0	78.2 / 2.1
300+CExt	75 / 2	63 / 3	83.9 / 1.3	83.2 / 1.4
300+RExt	62 / 4	50 / 5	81.4 / 1.4	79.9 / 2.1
300+RExt+CExt	75 / 2	63 / 3	85.2 / 1.1	84.4 / 1.2
2000+No	56 / 4	50 / 5	78.9 / 1.3	77.8 / 2.3
2000+CExt	72 / 2	63 / 3	86.6 / 0.7	83.2 / 1.4
2000+RExt	60 / 3	51 / 5	74.7 / 1.7	78.5 / 2.1
2000+RExt+CExt	72 / 2	64 / 3	82.6 / 1.1	84.4 / 1.3

Table 5: Costs and Payoffs of Reviewing the First 10 Generated Patches (excluding php)

Search Space	SPR	Prophet	Random (SPR)	Random (Prophet)
100+No	38 / 9	37 / 9	45.8 / 8.1	44.7 / 8.4
100+CExt	48 / 9	56 / 9	66.7 / 6.8	66.8 / 6.8
100+RExt	39 / 10	39 / 10	50.9 / 8.8	51.4 / 8.9
100+RExt+CExt	46 / 9	57 / 9	66.9 / 6.8	66.7 / 6.8
200+No	39 / 10	39 / 11	47.7 / 9.1	49.3 / 10.4
200+CExt	39 / 10	57 / 11	59.7 / 7.6	68.1 / 7.9
200+RExt	39 / 10	40 / 11	47.8 / 9.1	51.6 / 10.1
200+RExt+CExt	39 / 10	58 / 11	59.6 / 7.6	68.0 / 7.9
300+No	32 / 9	39 / 11	43.8 / 8.1	50.2 / 10.4
300+CExt	32 / 9	57 / 11	59.3 / 6.4	72.8 / 7.9
300+RExt	34 / 9	40 / 11	45.8 / 8.1	51.5 / 10.2
300+RExt+CExt	34 / 9	58 / 11	61.3 / 6.4	69.5 / 8.0
2000+No	7 / 5	25 / 7	16.7 / 4.4	37.4 / 6.3
2000+CExt	7 / 5	34 / 5	29.8 / 2.9	46.4 / 4.0
2000+RExt	9 / 5	17 / 6	18.7 / 4.4	29.3 / 5.3
2000+RExt+CExt	7 / 5	27 / 5	29.8 / 2.9	47.4 / 3.2

Table 6: Costs and Payoffs of Reviewing the First 10 Generated Patches (php only)

search space); with the largest evaluated search space (2000+RExt+CExt), Prophet times out for seven defects (whose correct patches are inside the search space).

Note that many previous systems [16, 38, 14, 28] neglect the weak test suite problem and do not evaluate whether the generated patches are correct or not. In contrast, our results show that the weak test suite problem is at least as important as the search space explosion problem. In fact, for all evaluated search space configurations, there are more defects for which SPR or Prophet generates plausible but incorrect patches than for which SPR or Prophet times out.

4.3 SPR and Prophet Effectiveness

We compare the effectiveness of the SPR and Prophet patch prioritization orders by measuring the costs and payoffs for a human developer who reviews the generated patches to find a correct patch. We consider a scenario in which the developer reviews the first 10 generated patches one by one for each defect until he finds a correct patch. He gives up if none of the first 10 patches are correct. For each system and each search space configuration, we compute (over the 24 defects that have correct patches in the full SPR and Prophet search space) 1) the total number of patches the developer reviews (this number is the cost) and 2) the total number of defects for which the developer obtains a correct patch (this number is the payoff). We also compute the expected costs and payoffs if the developer examines the generated plausible SPR and Prophet patches in a random order. The raw data used to compute these numbers is available at Appendix.

Tables 5 and 6 present these costs and payoffs. The first column presents the search space configuration. The second and third columns present the costs and payoffs for the SPR and Prophet patch prioritization orders; the fourth and fifth columns present the corresponding costs and payoffs for the random orders. Each entry is of the form X/Y, where X is the total number of patches that the developer reviews and Y is the total number of defects for which he obtains a correct patch. These numbers highlight the effectiveness of the SPR and Prophet patch prioritization in identifying correct patches within much larger sets of plausible but incorrect patches.

Table 6 presents the corresponding results for the php defects. These numbers highlight the difference that a stronger test suite can make in the success of finding correct patches. The correct patch selection probabilities are dramatically higher for php than for the other benchmarks. But note that as the patch search spaces become large, the number of defects for which the developer obtains correct patches become smaller, reflecting 1) the increasing inability of the systems to find any correct patch in the explored space within the 12 hour timeout and 2) the increasing presence of blocking plausible but incorrect patches.

Finally, these numbers highlight the effectiveness of the Prophet learned patch prioritization across the board — the developer always obtains correct patches for at least as many defects with Prophet as with SPR.

5. THREATS TO VALIDITY

This paper presents a systematic study of search space tradeoffs with SPR and Prophet. One threat to validity is that our results will not generalize to other benchmark sets and other patch generation systems. Note that the benchmark set was developed by other researchers, not by us, with the goal of obtaining a large, unbiased, and realistic benchmark set [16]. And this same benchmark set has been used to evaluate many previous patch generation systems [16, 38, 28, 29, 17]. The observations in this paper are consistent with previous results reported for other systems on this benchmark set [29, 16, 38, 28].

Another threat to validity is that stronger test suites will become the norm so that the results for benchmark applications other than php will not generalize to other applications. We note that 1) comprehensive test coverage is widely considered to be beyond reach for realistic applications, and 2) even php, which has by far the strongest test suite in the set of benchmark applications, has multiple defects for which the number of plausible patches exceeds the number of correct patches by one to two orders of magnitude.

6. RELATED WORK

ClearView: ClearView is a generate-and-validate system that observes normal executions to learn invariants that characterize safe behavior [27]. It deploys monitors that

detect crashes, illegal control transfers and out of bounds write defects. In response, it selects a nearby invariant that the input that triggered the defect violates, and generates patches that take a repair action to enforce the invariant.

A Red Team evaluation found that ClearView was able to automatically generate patches that eliminate 9 of 10 targeted Firefox vulnerabilities and enable Firefox to continue to execute successfully [27]. For 5 of these 9 defects, the first patch that ClearView generated was successful. For 1 of the 9, the second patch was successful. For 2 of the 9, the third patch was successful. The final defect that ClearView was able to patch required 3 distinct ClearView patches — each of the first 2 ClearView patches exposed a new defect that ClearView then patched. For these 3 defects, the first patch was successful. These numbers illustrate the density with which successful patches appear in the ClearView search space. We attribute this density, in part, to the fact that ClearView leverages the learned invariant information to focus the search on successful patches and does not rely solely on the validation test suite.

Kali: Kali is a generate-and-validate system that deploys a very simple strategy — it simply removes functionality. This strategy is obviously not intended to correctly repair a reasonable subset of the defects that occur in practice. Nevertheless, the results show that this strategy is at least as effective as previous more involved repair strategies that aspire to successfully repair a broad class of defects [29]. The standard scenario is that the defect occurs in a region of code that one of the negative test cases but few if any of the positive test cases exercises. Kali simply deletes the region of code to produce a program that passes the validation test suite but immediately fails on new test cases that exercise the removed functionality.

These results are consistent with the results presented in this paper, highlight the inadequacy of current test suites to successfully identify incorrect patches, and identify one prominent source of the relatively many plausible but incorrect patches that occur in current patch search spaces.

A recent paper [7] evaluates GenProg patches, NOPOL [3] patches, and Kali remove statement patches for 224 Java program defects in the Defects4J dataset [12]. The results are, in general, consistent with our results in this paper. Out of 42 manually analyzed plausible patches, the analysis indicates that only 8 patches are undoubtedly correct.

GenProg, AE, and RSRepair: GenProg [16], AE [38], and RSRepair [28] were all evaluated on (for RSRepair, a subset of) the same benchmark set that we use in this paper to evaluate the SPR and Prophet search spaces. The presented results in the GenProg, AE, and RSRepair papers [16, 38, 28] focus on the ability of the systems to generate plausible patches for the defects in the benchmark set. There is no attempt to determine whether the generated patches are correct or not. Unfortunately, the presented evaluations of these systems suffer from the fact that the testing infrastructure used to validate the candidate patches contains errors that cause the systems to incorrectly accept implausible patches that do not even pass all of the test cases in the validation test suite [29].

A subsequent study of these systems corrects these errors and sheds more light on the subject [29]. This study found that 1) the systems generate correct patches for only 2 (GenProg, RSRepair) or 3 (AE) of the 105 benchmark defects/functionality changes in this benchmark set, 2) the systems generate plausible but incorrect patches for 16 (GenProg), 8 (RSRepair), and 24 (AE) defects/functionality changes, and 3) the majority of the plausible patches, including all correct patches, are equivalent to a single modification that deletes functionality. Moreover, the correct SPR and Prophet patches for these defects lie outside the GenProg, AE, and RSRepair search space (suggesting that these systems will never be able to generate a correct patch for these defects) [17].

These results are broadly consistent with the results presented in this paper. The study found that only 5 of the 110 plausible GenProg patches are correct, only 4 of the analyzed plausible 44 RSRepair patches are correct, and only 3 of the 27 plausible AE patches are correct [29]. These results are consistent with the hypothesis that, in the GenProg, AE, and RSRepair search space, as in the SPR and Prophet search spaces, plausible but incorrect patches occur more abundantly than correct patches. A reasonable conclusion is that the GenProg, AE, and RSRepair search space, while containing fewer correct and plausible patches than the richer SPR and Prophet search spaces [17, 18], still exhibits the basic pattern of sparse correct patches and more abundant plausible but incorrect patches.

A subsequent study of GenProg and RSRepair (under the name TrpAutoRepair) on small student programs provides further support for this hypothesis [35]. The results indicate that patches validated on one test suite typically fail to generalize to produce correct results on other test suites not used to validate the patches. These results are consistent with a GenProg/RSRepair search space that contains sparse correct patches and more abundant plausible but incorrect patches.

We note that RSRepair uses random search; previous research found that the GenProg genetic search algorithm performs no better than random search on a subset of the benchmarks [28]. All of the published research is consistent with the hypothesis that, because of the relative abundance of plausible but incorrect patches, systems (such as GenProg and RSRepair) that perform no better than random search will need to incorporate additional sources of information other than the validation test suite if they are to successfully generate correct patches in the presence of current relatively weak test suites.[3]

NOPOL: NOPOL [3, 7] is an automatic repair tool focusing on branch conditions. It identifies branch statement directions that can pass negative test cases and then uses SMT solvers to generate repairs for the branch condition.

PAR: PAR [14] is another prominent automatic patch generation system. PAR is based on a set of predefined human-provided patch templates. We are unable to directly compare PAR with other systems because, despite repeated requests to the authors of the PAR paper over the course of 11 months, the authors never provided us with the patches that PAR was reported to have generated [14] or the necessary materials to would have enabled us to reproduce the PAR experiments.

[3]Of course, if the patch space does not contain correct patches, stronger test suites will prevent the system from generating any patches at all. This is the case for GenProg — at most two additional test cases per defect completely disable GenProg's ability to produce any patch at all except the five correct patches in its search space [29].

The PAR search space (with the eight templates in the PAR paper [14]) is in fact a subset of the SPR and Prophet search space. Monperrus found that PAR fixes the majority of its benchmark defects with only two templates ("Null Pointer Checker" and "Condition Expression Adder/Remover/Replacer") [22].

Plastic Surgery Hypothesis: Barr et al. [1] studied the platic surgery hypothesis, i.e., changes to a codebase contain snippets that already exist in the codebase at the time of the changes. The effectiveness of copy mutations of many patch generation systems relies on this hypothesis. Their results show that 10% of the commits can be completely constructed from existing code.

Fix Ingredient Availability: Martinez et. al. [21] studied more than 7,000 human commits in six open source programs to measure the fix ingredient availability, i.e., the percentage of commits that could be synthesized solely from existing lines of code. The results show that 3-17% of the commits can be synthesized from existing lines of code.

Real World Fix: Zhong and Su systematically analyzed more than 9000 real world bug fixes and studied several research questions including how many locations each bug fix modifies and what modification operators are essential for bug fixing [39].

Repair Model: Martinez and Monperrus manually mined previous human patches and suggest that if a patch generation system works with a non-uniform probabilistic model, the system would find plausible patches faster [20]. In contrast, Prophet [18] automatically learns a probabilistic model from past successful patches. Prophet is the first patch generation system to operate with such a learned model [18] to identify potentially correct patches.

SemFix and MintHint: SemFix [23] and MintHint [13] replace the faulty expression with a symbolic value and use symbolic execution techniques [2] to find a replacement expression that enables the program to pass all test cases. SemFix and MintHint are evaluated only on applications with less than 10000 lines of code. In addition, these techniques cannot generate fixes for statements with side effects.

CodePhage: Horizontal code transfer automatically locates correct code in one application, then transfers that code into another application [34]. This technique has been applied to eliminate otherwise fatal integer overflow, buffer overflow, and divide by zero errors and shows enormous potential for leveraging the combined talents and labor of software development efforts worldwide.

Defect Repair via Q&A Sites: Gao et. al. [10] propose to repair recurring defects by analyzing Q&A sites such as Stack Overflow. The proposed technique locates the relevant Q&A page for a recurring defect via a search engine query, extracts code snippets from the page, and renames variables in the extracted code snippets to generate patches. This technique addresses the weak test suite problem with the additional information collected from the Q&A sites.

AutoFixE: AutoFix-E [37, 26] operates with a set of fix schemas to repair Eiffel programs with human-supplied specifications called contracts.

Domain Specific Repair Generation: Other program repair systems include VEJOVIS [25] and Gopinath et al. [11], which applies domain specific techniques to repair DOM-related faults in JavaScript and selection statements in database programs respectively.

Failure-Oblivious Computing: Failure-oblivious computing [30] checks for out of bounds reads and writes. It discards out of bounds writes and manufactures values for out of bounds reads. This eliminates data corruption from out of bounds writes, eliminates crashes from out of bounds accesses, and enables the program to continue execution along its normal execution path.

RCV: RCV [19] enables applications to survive null dereference and divide by zero errors. It discards writes via null references, returns zero for reads via null references, and returns zero as the result of divides by zero. Execution continues along the normal execution path.

Bolt: Bolt [15] attaches to a running application, determines if the application is in an infinite loop, and, if so, exits the loop. A user can also use Bolt to exit a long-running loop. In both cases the goal is to enable the application to continue useful execution. Bolt uses checkpoint and restore to explore a search space of loop exit strategies, including jumping to the instruction immediately following the loop as well as returning to one of the (in general many) invoked procedures on the call stack.

Cyclic Memory Allocation: Cyclic memory allocation eliminates memory leaks by cyclically allocating objects out of a fixed-size buffer [24].

Data Structure Repair: Data structure repair enables applications to recover from data structure corruption errors [5, 6]. It enforces a data structure consistency specification. This specification can be provided by a developer or automatically inferred from correct program executions [4]. It is also possible to use violated assertions as a basis of reparing data structures [8].

In general, there are multiple repaired data structures that satisfy the consistency specification. Data structure repair operates over search space generated by a set of repair operations to generate a repaired data structure that is heuristically close to the original unrepaired data structure. The goal is not to obtain a hypothetical correct data structure, but instead a consistent data structure that enables acceptable continued execution.

Self-Stabilizing Java: Self-Stabilizing Java uses a type system to ensure that the impact of any errors are eventually flushed from the system, returning the system back to a consistent state [9].

Task Skipping and Loop Perforation: Task skipping [31, 32] discards task executions and loop perforation [33] discards loop iterations. The result is a generated search space of programs with varying performance and accuracy properties, some of which may be more desirable than the original. In addition to enabling programs to survive tasks or loop iterations that trigger otherwise fatal errors, task skipping and loop perforation can also improve performance.

7. CONCLUSION

The scope of any automatic patch generation system is limited by the range of patches it can search — to repair more defects, systems will need to work with spaces that contain more correct patches. This paper characterizes, for the first time, how larger and richer search spaces that contain more correct patches can, counterintuitively, hamper the ability of the system to find correct patches. It therefore identifies a key challenge that designers of future sys-

tems must overcome for their systems to successfully generate patches for broader classes of defects.

Experience with previous successful patch generation systems such as ClearView and Prophet highlights how leveraging information outside the validation test suite enables these systems to successfully identify the few correct patches within the many plausible patches (most of which are incorrect) that the test suite validates. We anticipate that future successful automatic patch generation systems will deploy even more sophisticated techniques that leverage the full range of available information (test suites, previous successful patches, documentation, even formal specifications) to successfully identify the correct patches available in larger, richer search spaces.

Acknowledgements

We thank the anonymous reviewers for their insightful comments on early draft of the paper. We thank Yu Lao for logistics support. This research was supported by DARPA (Grant FA8650-11-C-7192 and FA8750-14-2-0242).

8. REFERENCES

[1] E. T. Barr, Y. Brun, P. Devanbu, M. Harman, and F. Sarro. The Plastic Surgery Hypothesis. In *Proceedings of the 22nd ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE)*, pages 306–317, Hong Kong, China, November 2014.

[2] C. Cadar, D. Dunbar, and D. Engler. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 209–224, Berkeley, CA, USA, 2008. USENIX Association.

[3] F. DeMarco, J. Xuan, D. Le Berre, and M. Monperrus. Automatic repair of buggy if conditions and missing preconditions with smt. In *Proceedings of the 6th International Workshop on Constraints in Software Testing, Verification, and Analysis*, CSTVA 2014, pages 30–39, New York, NY, USA, 2014. ACM.

[4] B. Demsky, M. D. Ernst, P. J. Guo, S. McCamant, J. H. Perkins, and M. C. Rinard. Inference and enforcement of data structure consistency specifications. In *Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2006, Portland, Maine, USA, July 17-20, 2006*, pages 233–244, 2006.

[5] B. Demsky and M. C. Rinard. Automatic detection and repair of errors in data structures. In *OOPSLA*, pages 78–95, 2003.

[6] B. Demsky and M. C. Rinard. Goal-directed reasoning for specification-based data structure repair. *IEEE Trans. Software Eng.*, 32(12):931–951, 2006.

[7] T. Durieux, M. Martinez, M. Monperrus, R. Sommerard, and J. Xuan. Automatic repair of real bugs: An experience report on the defects4j dataset. *CoRR*, abs/1505.07002, 2015.

[8] B. Elkarablieh, I. Garcia, Y. L. Suen, and S. Khurshid. Assertion-based repair of complex data structures. In *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ASE '07', pages 64–73, New York, NY, USA, 2007. ACM.

[9] Y. H. Eom and B. Demsky. Self-stabilizing java. In *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '12, Beijing, China - June 11 - 16, 2012*, pages 287–298, 2012.

[10] Q. Gao, H. Zhang, J. Wang, Y. Xiong, L. Zhang, and H. Mei. Fixing recurring crash bugs via analyzing Q&A sites. In *Proc. of ASE*, 2015.

[11] D. Gopinath, S. Khurshid, D. Saha, and S. Chandra. Data-guided repair of selection statements. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 243–253, New York, NY, USA, 2014. ACM.

[12] R. Just, D. Jalali, and M. D. Ernst. Defects4j: a database of existing faults to enable controlled testing studies for java programs. In *International Symposium on Software Testing and Analysis, ISSTA '14, San Jose, CA, USA - July 21 - 26, 2014*, pages 437–440, 2014.

[13] S. Kaleeswaran, V. Tulsian, A. Kanade, and A. Orso. Minthint: Automated synthesis of repair hints. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 266–276, New York, NY, USA, 2014. ACM.

[14] D. Kim, J. Nam, J. Song, and S. Kim. Automatic patch generation learned from human-written patches. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13', pages 802–811. IEEE Press, 2013.

[15] M. Kling, S. Misailovic, M. Carbin, and M. Rinard. Bolt: on-demand infinite loop escape in unmodified binaries. In *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, OOPSLA '12', pages 431–450. ACM, 2012.

[16] C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each. In *Proceedings of the 2012 International Conference on Software Engineering*, ICSE 2012, pages 3–13. IEEE Press, 2012.

[17] F. Long and M. Rinard. Staged program repair with condition synthesis. In *Proceedings of 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2015) Proceedings of ESEC/FSE*, 2015.

[18] F. Long and M. Rinard. Automatic patch generation by learning correct code. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL 2016, pages 298–312, New York, NY, USA, 2016. ACM.

[19] F. Long, S. Sidiroglou-Douskos, and M. Rinard. Automatic runtime error repair and containment via recovery shepherding. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14', pages 227–238, New York, NY, USA, 2014. ACM.

[20] M. Martinez and M. Monperrus. Mining software repair models for reasoning on the search space of automated program fixing. *Empirical Software Engineering*, 20(1):176–205, 2015.

[21] M. Martinez, W. Weimer, and M. Monperrus. Do the fix ingredients already exist? an empirical inquiry into the redundancy assumptions of program repair approaches. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 492–495, New York, NY, USA, 2014. ACM.

[22] M. Monperrus. A critical review of "automatic patch generation learned from human-written patches": Essay on the problem statement and the evaluation of automatic software repair. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 234–242, New York, NY, USA, 2014. ACM.

[23] H. D. T. Nguyen, D. Qi, A. Roychoudhury, and S. Chandra. Semfix: Program repair via semantic analysis. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13', pages 772–781, Piscataway, NJ, USA, 2013. IEEE Press.

[24] H. H. Nguyen and M. C. Rinard. Detecting and eliminating memory leaks using cyclic memory allocation. In *Proceedings of the 6th International Symposium on Memory Management, ISMM 2007, Montreal, Quebec, Canada, October 21-22, 2007*, pages 15–30, 2007.

[25] F. S. Ocariza, Jr., K. Pattabiraman, and A. Mesbah. Vejovis: Suggesting fixes for javascript faults. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 837–847, New York, NY, USA, 2014. ACM.

[26] Y. Pei, C. A. Furia, M. Nordio, Y. Wei, B. Meyer, and A. Zeller. Automated fixing of programs with contracts. *IEEE Trans. Software Eng.*, 40(5):427–449, 2014.

[27] J. H. Perkins, S. Kim, S. Larsen, S. Amarasinghe, J. Bachrach, M. Carbin, C. Pacheco, F. Sherwood, S. Sidiroglou, G. Sullivan, W.-F. Wong, Y. Zibin, M. D. Ernst, and M. Rinard. Automatically patching errors in deployed software. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 87–102. ACM, 2009.

[28] Y. Qi, X. Mao, Y. Lei, Z. Dai, and C. Wang. The strength of random search on automated program repair. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 254–265, New York, NY, USA, 2014. ACM.

[29] Z. Qi, F. Long, S. Achour, and M. Rinard. An analysis of patch plausibility and correctness for generate-and-validate patch generation systems. In *Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2015*, 2015.

[30] M. Rinard, C. Cadar, D. Dumitran, D. M. Roy, T. Leu, and W. S. Beebee. Enhancing server availability and security through failure-oblivious computing. In *OSDI*, pages 303–316, 2004.

[31] M. C. Rinard. Probabilistic accuracy bounds for fault-tolerant computations that discard tasks. In *ICS*, pages 324–334, 2006.

[32] M. C. Rinard. Using early phase termination to eliminate load imbalances at barrier synchronization points. In *OOPSLA*, pages 369–386, 2007.

[33] S. Sidiroglou, S. Misailovic, H. Hoffmann, and M. Rinard. Managing performance vs. accuracy trade-offs with loop perforation. FSE, 2011.

[34] S. Sidiroglou-Douskos, E. Lahtinen, F. Long, and M. Rinard. Automatic error elimination by horizontal code transfer across multiple applications. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, Portland, OR, USA, June 15-17, 2015*, pages 43–54, 2015.

[35] E. K. Smith, E. Barr, C. Le Goues, and Y. Brun. Is the Cure Worse than the Disease? Overfitting in Automated Program Repair. In *Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 532–543, Bergamo, Italy, September 2015.

[36] S. Son, K. S. McKinley, and V. Shmatikov. Fix me up: Repairing access-control bugs in web applications. In *NDSS*, 2013.

[37] Y. Wei, Y. Pei, C. A. Furia, L. S. Silva, S. Buchholz, B. Meyer, and A. Zeller. Automated fixing of programs with contracts. In *Proceedings of the 19th International Symposium on Software Testing and Analysis*, ISSTA '10', pages 61–72, New York, NY, USA, 2010. ACM.

[38] W. Weimer, Z. P. Fry, and S. Forrest. Leveraging program equivalence for adaptive program repair: Models and first results. In *ASE'13*, pages 356–366, 2013.

[39] H. Zhong and Z. Su. An empirical study on real bug fixes. In *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*, pages 913–923, 2015.

APPENDIX

Tables 7-38 present the detailed experimental results. Each table presents the results of SPR or Prophet on one search space configuration. The first column of the table presents the defect id. The second column of the table presents the total number of candidate patch templates in the search space. The third column presents the number of patch templates that manipulate branch conditions. The fourth column presents the total number of evaluated patch templates in 12 hours. The fifth column presents the total number of evaluated condition patch templates during in 12 hours. The sixth column presents the number of templates for which generate plausible patches. The seventh column presents the number of condition templates for which generate plausible patches. The eighth column presents the total number of plausible patches the system finds in 12 hours. The ninth column presents the number of plausible patches which manipulate branch conditions. The tenth column presents the number of correct patches the system finds in 12 hours. The eleventh column presents the rank of the template that generates the first correct patch in the search space. The twelfth column presents the rank of the template among plausible templates. The last column presents the rank of the correct patch among all generated plausible patches.

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	10949	3394	10949	3394	33	28	111	106	0	-	-	-
libtiff-5b0217-3dfb33	83937	19298	52303	16953	208	147	211	150	0	-	-	-
libtiff-d13be7-ccadf4	184068	141307	63046	58498	1423	1423	1703	1423	1	1093	1	1
lighttpd-2661-2662	34688	27541	34688	27541	70	8	73	11	0	-	-	-
lighttpd-1913-1914	32903	21707	32903	21707	4	4	4	4	0	-	-	-
python-69934-69935	17828	5874	6501	4383	0	0	0	0	0	-	-	-
gmp-13420-13421	29703	5319	29703	5319	0	0	0	0	0	-	-	-
gzip-a1d3d4-f17cbd	25303	7535	25303	7535	5	0	5	0	1	1135	1	1
python-70056-70059	22537	9353	4106	3644	0	0	0	0	0	-	-	-
fbc-5458-5459	4864	2144	4864	2144	26	26	46	35	2	19	1	1
libtiff-ee2ce5-b5691a	34863	25186	34863	25186	328	328	328	328	1	110	1	1
php-310991-310999	47688	11762	21206	9962	1	1	1	1	1	808	1	1
php-308734-308761	14	11	14	11	0	0	0	0	0	-	-	-
php-308262-308315	22003	1869	5283	1869	0	0	0	0	0	150	-	-
php-307562-307561	21252	4869	13996	4869	1	0	1	0	1	2221	1	1
php-309579-309580	41568	6890	19370	6890	2	2	2	2	1	288	1	1
php-310011-310050	34470	8642	2394	2082	63	13	69	23	1	640	13	21
php-309688-309716	47600	11980	2609	2540	71	71	95	95	1	2740	64	86
php-309516-309535	19142	4578	19142	4578	1	0	1	0	1	8851	1	1
php-307846-307853	13971	3009	13971	3009	1	0	1	0	1	8470	1	1
php-311346-311348	8096	3027	8096	3027	50	38	72	60	2	25	1	1
php-307914-307915	25707	7731	25707	7731	1	0	1	0	1	1	1	1
php-309111-309159	32541	6583	32541	6583	10	1	10	1	1	6838	9	9
php-309892-309910	8665	1432	8665	1432	21	17	26	22	4	161	1	1

Table 7: Prophet-100 Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	10949	3394	10949	3394	29	24	56	51	0	-	-	-
libtiff-5b0217-3dfb33	83937	19298	1249	1112	146	146	3192	3192	0	-	-	-
libtiff-d13be7-ccadf4	184068	141307	3938	3896	116	116	4540	4362	2	1093	1	1
lighttpd-2661-2662	34688	27541	34688	27541	78	16	109	47	0	-	-	-
lighttpd-1913-1914	32903	21707	32903	21707	4	4	12	12	0	-	-	-
python-69934-69935	17828	5874	6501	4383	0	0	0	0	0	-	-	-
gmp-13420-13421	29703	5319	29703	5319	0	0	0	0	0	-	-	-
gzip-a1d3d4-f17cbd	25303	7535	25303	7535	5	0	5	0	1	1135	1	1
python-70056-70059	22537	9353	4106	3644	0	0	0	0	0	-	-	-
fbc-5458-5459	4864	2144	4864	2144	23	23	98	71	0	19	-	-
libtiff-ee2ce5-b5691a	34863	25186	4576	4282	113	113	3889	3889	1	110	1	1
php-310991-310999	47688	11762	20995	9962	1	1	1	1	1	808	1	1
php-308734-308761	14	11	14	11	0	0	0	0	0	-	-	-
php-308262-308315	22003	1869	2019	1510	18	18	44	44	1	150	2	2
php-307562-307561	21252	4869	13736	4869	1	0	1	0	1	2221	1	1
php-309579-309580	41568	6890	15676	6890	11	11	38	38	1	288	6	12
php-310011-310050	34470	8642	2424	2082	71	7	76	14	1	640	7	9
php-309688-309716	47600	11980	538	501	2	2	96	96	0	2740	-	-
php-309516-309535	19142	4578	19142	4578	1	0	1	0	1	8851	1	1
php-307846-307853	13971	3009	13971	3009	1	0	1	0	1	8470	1	1
php-311346-311348	8096	3027	1485	1397	5	5	101	101	1	25	1	1
php-307914-307915	25707	7731	25707	7731	1	0	1	0	1	1	1	1
php-309111-309159	32541	6583	32541	6583	10	1	10	1	1	6838	9	9
php-309892-309910	8665	1432	451	345	11	11	96	96	1	161	2	12

Table 8: Prophet-100-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	12718	3394	12718	3394	33	28	111	106	0	-	-	-
libtiff-5b0217-3dfb33	88613	19298	53568	16729	210	147	213	150	0	-	-	-
libtiff-d13be7-ccadf4	186777	141307	60747	54316	1185	1185	1405	1185	1	1202	1	1
lighttpd-2661-2662	35810	27541	35810	27541	66	4	69	7	0	-	-	-
lighttpd-1913-1914	34917	21707	34917	21707	4	4	4	4	0	-	-	-
python-69934-69935	18168	5874	6408	4339	0	0	0	0	0	-	-	-
gmp-13420-13421	49743	5319	49743	5319	0	0	0	0	0	-	-	-
gzip-a1d3d4-f17cbd	31251	7535	31251	7535	5	0	5	0	1	715	1	1
python-70056-70059	26398	9353	3950	3517	0	0	0	0	0	-	-	-
fbc-5458-5459	5938	2144	5938	2144	28	26	48	35	2	16	1	1
libtiff-ee2ce5-b5691a	37661	25186	37661	25186	328	328	328	328	1	149	1	1
php-310991-310999	48843	11762	19316	8758	1	1	1	1	1	1509	1	1
php-308734-308761	14	11	14	11	0	0	0	0	0	-	-	-
php-308262-308315	22472	1869	2963	1648	4	3	6	5	1	206	1	1
php-307562-307561	21944	4869	14071	4869	1	0	1	0	1	2962	1	1
php-309579-309580	42322	6890	19229	6890	2	2	2	2	1	330	1	1
php-310011-310050	35256	8642	4112	3411	60	13	70	23	1	742	9	14
php-309688-309716	48118	11980	3019	2899	68	68	92	92	0	6827	-	-
php-309516-309535	21008	4578	21008	4578	1	0	1	0	1	9788	1	1
php-307846-307853	16435	3009	16435	3009	1	0	1	0	1	10250	1	1
php-311346-311348	8173	3027	8173	3027	51	38	73	60	2	17	1	1
php-307914-307915	27121	7731	27121	7731	1	0	1	0	1	1	1	1
php-309111-309159	34080	6583	34080	6583	10	1	10	1	1	8261	10	10
php-309892-309910	10855	1432	3184	1368	64	17	69	22	4	313	1	1

Table 9: Prophet-100-RExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	12718	3394	12718	3394	29	24	56	51	0	-	-	-
libtiff-5b0217-3dfb33	88613	19298	1455	1325	146	146	3192	3192	0	-	-	-
libtiff-d13be7-ccadf4	186777	141307	4252	4207	115	115	4525	4347	2	1202	1	1
lighttpd-2661-2662	35810	27541	35810	27541	74	12	97	35	0	-	-	-
lighttpd-1913-1914	34917	21707	34917	21707	4	4	12	12	0	-	-	-
python-69934-69935	18168	5874	6402	4339	0	0	0	0	0	-	-	-
gmp-13420-13421	49743	5319	49743	5319	0	0	0	0	0	-	-	-
gzip-a1d3d4-f17cbd	31251	7535	31251	7535	5	0	5	0	1	715	1	1
python-70056-70059	26398	9353	3967	3517	0	0	0	0	0	-	-	-
fbc-5458-5459	5938	2144	5938	2144	25	23	100	71	0	16	-	-
libtiff-ee2ce5-b5691a	37661	25186	5031	4747	113	113	3608	3608	1	149	1	1
php-310991-310999	48843	11762	20496	9251	1	1	1	1	1	1509	1	1
php-308734-308761	14	11	14	11	0	0	0	0	0	-	-	-
php-308262-308315	22472	1869	2163	1526	18	18	43	43	1	206	2	2
php-307562-307561	21944	4869	14132	4869	1	0	1	0	1	2962	1	1
php-309579-309580	42322	6890	15022	6890	11	11	38	38	1	330	6	12
php-310011-310050	35256	8642	4122	3411	58	7	64	14	1	742	7	9
php-309688-309716	48118	11980	920	857	2	2	92	92	0	6827	-	-
php-309516-309535	21008	4578	21008	4578	1	0	1	0	1	9788	1	1
php-307846-307853	16435	3009	16435	3009	1	0	1	0	1	10250	1	1
php-311346-311348	8173	3027	1462	1363	5	5	106	106	1	17	1	1
php-307914-307915	27121	7731	27121	7731	1	0	1	0	1	1	1	1
php-309111-309159	34080	6583	34080	6583	10	1	10	1	1	8261	10	10
php-309892-309910	10855	1432	470	372	11	11	100	100	1	313	2	12

Table 10: Prophet-100-RExt-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	39287	14161	39287	14161	33	28	111	106	0	-	-	-
libtiff-5b0217-3dfb33	221134	65017	64363	31834	247	149	250	152	0	50770	-	-
libtiff-d13be7-ccadf4	296426	228659	70050	65731	1423	1423	1703	1423	1	1183	1	1
lighttpd-2661-2662	120263	98028	79882	67844	48	2	50	4	0	-	-	-
lighttpd-1913-1914	68199	48133	37214	30701	58	58	58	58	0	-	-	-
python-69934-69935	47544	13775	6546	4750	0	0	0	0	0	-	-	-
gmp-13420-13421	50672	8763	50672	8763	3	0	3	0	2	14102	1	1
gzip-a1d3d4-f17cbd	48702	15104	48702	15104	14	0	14	0	1	1929	1	1
python-70056-70059	39599	17961	4032	3645	0	0	0	0	0	-	-	-
fbc-5458-5459	9857	4495	9857	4495	27	27	47	36	2	33	1	1
libtiff-ee2ce5-b5691a	171379	106867	137262	103332	328	328	328	328	1	280	1	1
php-310991-310999	89230	18988	21250	11637	1	1	1	1	1	907	1	1
php-308734-308761	14692	4160	14692	4160	4	4	4	4	2	5376	1	1
php-308262-308315	90431	10845	8496	7508	3	3	5	5	1	1365	1	1
php-307562-307561	31597	6997	14698	6997	1	0	1	0	1	2672	1	1
php-309579-309580	60351	11416	23605	11416	2	2	2	2	1	767	1	1
php-310011-310050	77671	16558	4857	4556	63	13	69	23	1	1348	13	21
php-309688-309716	71633	15744	3310	3241	68	68	92	92	1	3465	61	83
php-309516-309535	27098	6314	27098	6314	1	0	1	0	1	10954	1	1
php-307846-307853	22131	4757	16871	4709	1	0	1	0	1	10742	1	1
php-311346-311348	9799	3879	9799	3879	50	38	72	60	2	27	1	1
php-307914-307915	47988	15066	35684	15066	1	0	1	0	1	1	1	1
php-309111-309159	52908	12232	36533	11928	10	1	10	1	1	7701	9	9
php-309892-309910	40758	9999	13614	7118	21	17	26	22	4	462	1	1

Table 11: Prophet-200 Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	39287	14161	39287	14161	29	24	56	51	0	-	-	-
libtiff-5b0217-3dfb33	221134	65017	2054	1940	146	146	2776	2776	0	50770	-	-
libtiff-d13be7-ccadf4	296426	228659	4679	4657	111	111	4308	4191	2	1183	1	1
lighttpd-2661-2662	120263	98028	77400	65714	55	10	75	30	0	-	-	-
lighttpd-1913-1914	68199	48133	22474	19567	56	56	160	160	0	-	-	-
python-69934-69935	47544	13775	6616	4760	0	0	0	0	0	-	-	-
gmp-13420-13421	50672	8763	50672	8763	3	0	3	0	2	14102	1	1
gzip-a1d3d4-f17cbd	48702	15104	48702	15104	14	0	14	0	1	1929	1	1
python-70056-70059	39599	17961	4464	4074	0	0	0	0	0	-	-	-
fbc-5458-5459	9857	4495	6132	3920	34	34	118	89	0	33	-	-
libtiff-ee2ce5-b5691a	171379	106867	8258	7714	113	113	3458	3458	1	280	1	1
php-310991-310999	89230	18988	21436	11684	1	1	1	1	1	907	1	1
php-308734-308761	14692	4160	14692	4160	4	4	4	4	2	5376	1	1
php-308262-308315	90431	10845	7641	7304	15	15	36	36	1	1365	2	2
php-307562-307561	31597	6997	15277	6997	1	0	1	0	1	2672	1	1
php-309579-309580	60351	11416	18816	11416	11	11	38	38	1	767	6	11
php-310011-310050	77671	16558	4867	4556	61	7	69	14	1	1348	7	9
php-309688-309716	71633	15744	538	501	2	2	93	93	0	3465	-	-
php-309516-309535	27098	6314	27098	6314	1	0	1	0	1	10954	1	1
php-307846-307853	22131	4757	16927	4709	1	0	1	0	1	10742	1	1
php-311346-311348	9799	3879	1799	1719	5	5	103	103	1	27	1	1
php-307914-307915	47988	15066	35979	15066	1	0	1	0	1	1	1	1
php-309111-309159	52908	12232	36614	11949	10	1	10	1	1	7701	9	9
php-309892-309910	40758	9999	1417	1316	11	11	89	89	1	462	2	10

Table 12: Prophet-200-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	61171	14161	53629	13793	33	28	111	106	0	-	-	-
libtiff-5b0217-3dfb33	278857	65017	52657	22730	377	149	380	152	0	70740	-	-
libtiff-d13be7-ccadf4	300745	228659	70497	63153	819	819	939	819	1	1321	1	1
lighttpd-2661-2662	124449	98028	78470	64364	62	0	62	0	0	-	-	-
lighttpd-1913-1914	70712	48133	35590	27952	54	54	54	54	0	-	-	-
python-69934-69935	48560	13775	6699	4826	0	0	0	0	0	24090	-	-
gmp-13420-13421	79763	8763	64198	8713	1	0	1	0	1	39181	1	1
gzip-a1d3d4-f17cbd	58432	15104	58432	15104	14	0	14	0	1	1303	1	1
python-70056-70059	43598	17961	4428	4021	0	0	0	0	0	-	-	-
fbc-5458-5459	11828	4495	9775	4353	30	28	50	37	2	24	1	1
libtiff-ee2ce5-b5691a	190629	106867	119616	81875	328	328	328	328	1	408	1	1
php-310991-310999	90936	18988	21023	11071	1	1	1	1	1	1778	1	1
php-308734-308761	18784	4160	18784	4160	4	4	4	4	2	6242	1	1
php-308262-308315	92459	10845	8606	7477	4	3	6	5	1	1817	1	1
php-307562-307561	32546	6997	15557	6997	1	0	1	0	1	3481	1	1
php-309579-309580	61407	11416	22933	11416	2	2	2	2	1	603	1	1
php-310011-310050	78981	16558	6593	5934	52	24	66	41	1	1413	9	14
php-309688-309716	73424	15744	2177	2081	68	68	92	92	0	8488	-	-
php-309516-309535	29514	6314	29514	6314	1	0	1	0	1	11765	1	1
php-307846-307853	25447	4757	18938	4661	1	0	1	0	1	12791	1	1
php-311346-311348	9978	3879	10051	3879	49	38	71	60	2	20	1	1
php-307914-307915	50442	15066	36291	15066	1	0	1	0	1	1	1	1
php-309111-309159	58903	12232	34193	11196	10	1	10	1	1	9220	10	10
php-309892-309910	52975	9999	7971	4457	35	17	40	22	4	938	1	1

Table 13: Prophet-200-RExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	61171	14161	55865	14057	29	24	56	51	0	-	-	-
libtiff-5b0217-3dfb33	278857	65017	2032	1923	146	146	2806	2806	0	70740	-	-
libtiff-d13be7-ccadf4	300745	228659	4825	4803	109	109	4279	4162	2	1321	1	1
lighttpd-2661-2662	124449	98028	77686	64125	64	10	81	27	0	-	-	-
lighttpd-1913-1914	70712	48133	23341	19172	55	55	155	155	0	-	-	-
python-69934-69935	48560	13775	6699	4826	0	0	0	0	0	24090	-	-
gmp-13420-13421	79763	8763	63696	8713	1	0	1	0	1	39181	1	1
gzip-a1d3d4-f17cbd	58432	15104	58432	15104	14	0	14	0	1	1303	1	1
python-70056-70059	43598	17961	4385	3978	0	0	0	0	0	-	-	-
fbc-5458-5459	11828	4495	8182	4060	29	27	107	78	0	24	-	-
libtiff-ee2ce5-b5691a	190629	106867	6895	6520	113	113	3429	3429	1	408	1	1
php-310991-310999	90936	18988	20884	11014	1	1	1	1	1	1778	1	1
php-308734-308761	18784	4160	18784	4160	4	4	4	4	2	6242	1	1
php-308262-308315	92459	10845	7422	7082	15	15	36	36	1	1817	2	2
php-307562-307561	32546	6997	15557	6997	1	0	1	0	1	3481	1	1
php-309579-309580	61407	11416	17325	11416	11	11	38	38	1	603	6	11
php-310011-310050	78981	16558	6623	5934	54	12	61	19	1	1413	7	9
php-309688-309716	73424	15744	920	857	2	2	91	91	0	8488	-	-
php-309516-309535	29514	6314	29514	6314	1	0	1	0	1	11765	1	1
php-307846-307853	25447	4757	18964	4661	1	0	1	0	1	12791	1	1
php-311346-311348	9978	3879	1733	1643	5	5	105	105	1	20	1	1
php-307914-307915	50442	15066	36465	15066	1	0	1	0	1	1	1	1
php-309111-309159	58903	12232	33503	11036	10	1	10	1	1	9220	10	10
php-309892-309910	52975	9999	1413	1305	11	11	84	84	1	938	2	10

Table 14: Prophet-200-RExt-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	59327	22055	49749	21743	33	28	111	106	0	-	-	-
libtiff-5b0217-3dfb33	276142	85251	64841	32860	225	149	228	152	0	52950	-	-
libtiff-d13be7-ccadf4	568172	432083	76111	71854	1423	1423	1703	1423	1	1453	1	1
lighttpd-2661-2662	191579	159862	92356	80355	48	3	51	6	0	-	-	-
lighttpd-1913-1914	159739	114667	36479	30861	63	63	63	63	0	-	-	-
python-69934-69935	65326	18300	7258	5562	0	0	0	0	0	-	-	-
gmp-13420-13421	69661	12794	61424	12794	9	5	9	5	2	14989	6	6
gzip-a1d3d4-f17cbd	82349	18024	82349	18024	14	0	14	0	1	2250	1	1
python-70056-70059	58530	26327	4453	4155	0	0	0	0	0	-	-	-
fbc-5458-5459	16353	5416	10124	5416	28	28	48	37	2	33	1	1
libtiff-ee2ce5-b5691a	218252	142593	151378	118052	328	328	328	328	1	308	1	1
php-310991-310999	139311	31061	27055	18568	1	1	1	1	1	1396	1	1
php-308734-308761	30623	7437	13609	7432	4	4	4	4	2	7684	1	1
php-308262-308315	150963	19358	12615	11830	2	2	4	4	1	2046	1	1
php-307562-307561	60507	15748	18133	10420	1	0	1	0	1	4780	1	1
php-309579-309580	101940	17784	24749	17625	2	2	2	2	1	875	1	1
php-310011-310050	105542	23824	7441	7000	55	14	64	25	1	2203	14	23
php-309688-309716	95206	19747	528	501	68	68	90	90	1	3758	61	82
php-309516-309535	52377	12093	35403	11940	1	0	1	0	1	12002	1	1
php-307846-307853	40272	8051	16704	6229	3	2	4	3	1	12101	1	1
php-311346-311348	14543	5619	8465	5526	50	41	72	63	2	28	1	1
php-307914-307915	64378	20107	36206	18364	1	0	1	0	1	1	1	1
php-309111-309159	86627	22335	34642	13196	10	1	10	1	1	8749	9	9
php-309892-309910	62347	19484	16588	10223	21	17	26	22	4	532	1	1

Table 15: Prophet-300 Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	59327	22055	51701	21934	29	24	56	51	0	-	-	-
libtiff-5b0217-3dfb33	276142	85251	2433	2326	146	146	2794	2794	0	52950	-	-
libtiff-d13be7-ccadf4	568172	432083	10788	10704	108	108	4237	4120	2	1453	1	1
lighttpd-2661-2662	191579	159862	90638	78887	59	14	80	35	0	-	-	-
lighttpd-1913-1914	159739	114667	22555	20308	57	57	160	160	0	42256	-	-
python-69934-69935	65326	18300	7307	5579	0	0	0	0	0	-	-	-
gmp-13420-13421	69661	12794	60776	12794	7	3	22	18	2	14989	4	19
gzip-a1d3d4-f17cbd	82349	18024	82349	18024	14	0	14	0	1	2250	1	1
python-70056-70059	58530	26327	4507	4200	0	0	0	0	0	1495	-	-
fbc-5458-5459	16353	5416	6663	4350	34	34	118	89	0	33	-	-
libtiff-ee2ce5-b5691a	218252	142593	9848	9310	113	113	3467	3467	1	308	1	1
php-310991-310999	139311	31061	27584	18695	1	1	1	1	1	1396	1	1
php-308734-308761	30623	7437	13646	7432	4	4	4	4	2	7684	1	1
php-308262-308315	150963	19358	12072	11728	10	10	26	26	1	2046	2	2
php-307562-307561	60507	15748	18148	10436	1	0	1	0	1	4780	1	1
php-309579-309580	101940	17784	18107	14178	11	11	38	38	1	875	6	12
php-310011-310050	105542	23824	7451	7000	57	7	61	14	1	2203	7	9
php-309688-309716	95206	19747	528	501	2	2	90	90	0	3758	-	-
php-309516-309535	52377	12093	35698	11940	1	0	1	0	1	12002	1	1
php-307846-307853	40272	8051	15952	6206	5	4	12	11	1	12101	1	1
php-311346-311348	14543	5619	1923	1843	8	8	99	99	1	28	1	1
php-307914-307915	64378	20107	36360	18402	1	0	1	0	1	1	1	1
php-309111-309159	86627	22335	35676	13501	10	1	10	1	1	8749	9	9
php-309892-309910	62347	19484	1925	1845	11	11	81	81	1	532	2	10

Table 16: Prophet-300-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	95751	22055	43517	13623	31	28	109	106	0	62583	-	-
libtiff-5b0217-3dfb33	336808	85251	53883	23725	303	149	306	152	0	72938	-	-
libtiff-d13be7-ccadf4	573714	432083	76021	68438	556	556	609	556	1	1608	1	1
lighttpd-2661-2662	197676	159862	90518	76321	58	4	61	7	0	-	-	-
lighttpd-1913-1914	168946	114667	36761	29588	60	60	60	60	0	-	-	-
python-69934-69935	66616	18300	7338	5570	0	0	0	0	0	26404	-	-
gmp-13420-13421	107650	12794	65759	12094	9	5	9	5	1	40564	6	6
gzip-a1d3d4-f17cbd	100652	18024	102416	18024	14	0	14	0	1	1588	1	1
python-70056-70059	64192	26327	4479	4184	0	0	0	0	0	-	-	-
fbc-5458-5459	18782	5416	9032	4753	32	30	52	39	2	24	1	1
libtiff-ee2ce5-b5691a	241707	142593	121791	85492	328	328	328	328	1	444	1	1
php-310991-310999	142164	31061	26773	17940	1	1	1	1	1	2939	1	1
php-308734-308761	36012	7437	14040	6893	4	4	4	4	2	8317	1	1
php-308262-308315	159696	19358	12721	11808	2	2	4	4	1	2694	1	1
php-307562-307561	64392	15748	17926	10160	1	0	1	0	1	5763	1	1
php-309579-309580	103151	17784	24041	16881	2	2	2	2	1	749	1	1
php-310011-310050	107479	23824	9297	8378	44	25	62	43	1	2233	10	16
php-309688-309716	97372	19747	900	857	68	68	88	88	0	9296	-	-
php-309516-309535	56549	12093	35228	11710	1	0	1	0	1	12585	1	1
php-307846-307853	44992	8051	19506	6176	3	2	4	3	1	14098	1	1
php-311346-311348	14889	5619	8610	5526	50	41	72	63	2	21	1	1
php-307914-307915	67286	20107	36344	17795	1	0	1	0	1	1	1	1
php-309111-309159	102384	22335	32495	12321	10	1	10	1	1	10350	10	10
php-309892-309910	74780	19484	11247	7175	28	17	33	22	4	1194	1	1

Table 17: Prophet-300-RExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	95751	22055	45224	14092	27	24	54	51	0	62583	-	-
libtiff-5b0217-3dfb33	336808	85251	2369	2261	146	146	2848	2848	0	72938	-	-
libtiff-d13be7-ccadf4	573714	432083	11655	11585	107	107	4205	4088	2	1608	1	1
lighttpd-2661-2662	197676	159862	90748	76551	57	3	59	5	0	-	-	-
lighttpd-1913-1914	168946	114667	23858	20291	54	54	149	149	0	46828	-	-
python-69934-69935	66616	18300	7948	5824	0	0	0	0	0	26404	-	-
gmp-13420-13421	107650	12794	65438	12094	4	3	19	18	1	40564	4	19
gzip-a1d3d4-f17cbd	100652	18024	102416	18024	14	0	14	0	1	1588	1	1
python-70056-70059	64192	26327	4479	4184	0	0	0	0	0	1570	-	-
fbc-5458-5459	18782	5416	7041	4022	33	31	113	84	0	24	-	-
libtiff-ee2ce5-b5691a	241707	142593	8496	8118	113	113	3425	3425	1	444	1	1
php-310991-310999	142164	31061	26306	17900	1	1	1	1	1	2939	1	1
php-308734-308761	36012	7437	13969	6893	4	4	4	4	2	8317	1	1
php-308262-308315	159696	19358	11843	11506	10	10	25	25	1	2694	2	2
php-307562-307561	64392	15748	17976	10160	1	0	1	0	1	5763	1	1
php-309579-309580	103151	17784	17004	12979	11	11	38	38	1	749	6	12
php-310011-310050	107479	23824	9327	8378	50	12	60	19	1	2233	7	9
php-309688-309716	97372	19747	900	857	2	2	88	88	0	9296	-	-
php-309516-309535	56549	12093	34931	11710	1	0	1	0	1	12585	1	1
php-307846-307853	44992	8051	19371	6176	4	3	5	4	1	14098	1	1
php-311346-311348	14889	5619	1883	1795	8	8	102	102	1	21	1	1
php-307914-307915	67286	20107	36699	17978	1	0	1	0	1	1	1	1
php-309111-309159	102384	22335	32398	12291	10	1	10	1	1	10350	10	10
php-309892-309910	74780	19484	1925	1845	11	11	86	86	1	1194	2	10

Table 18: Prophet-300-RExt-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	448318	141182	43569	22779	31	28	109	106	0	-	-	-
libtiff-5b0217-3dfb33	1941177	1352092	83094	59078	182	149	185	152	0	91329	-	-
libtiff-d13be7-ccadf4	2353240	1771135	113514	106810	416	416	431	416	1	2820	1	1
lighttpd-2661-2662	1269307	1016413	120469	107039	46	10	54	18	0	1197010	-	-
lighttpd-1913-1914	1183286	935121	59987	56385	51	51	51	51	0	-	-	-
python-69934-69935	863452	386935	39078	29758	0	0	0	0	0	-	-	-
gmp-13420-13421	389124	104714	60147	20342	8	5	8	5	1	22027	6	6
gzip-a1d3d4-f17cbd	341681	92972	137970	43616	16	0	16	0	2	4299	1	1
python-70056-70059	526778	236707	4555	4551	0	0	0	0	0	-	-	-
fbc-5458-5459	16353	5416	8882	5169	28	28	47	36	2	33	1	1
libtiff-ee2ce5-b5691a	3237374	2609366	145761	129703	328	328	328	328	1	2601	1	1
php-310991-310999	783770	185001	56088	55384	1	1	1	1	1	8294	1	1
php-308734-308761	440926	151934	43304	38805	0	0	0	0	0	45726	-	-
php-308262-308315	637558	192659	57001	56034	1	1	2	2	1	8304	1	1
php-307562-307561	637588	163053	59252	53402	1	0	1	0	1	34355	1	1
php-309579-309580	621880	198376	65289	59970	2	2	2	2	1	7010	1	1
php-310011-310050	582198	162849	653	653	0	0	0	0	0	13331	-	-
php-309688-309716	609953	191952	30720	30659	34	34	35	35	0	34218	-	-
php-309516-309535	496992	136030	33781	27530	0	0	0	0	0	41978	-	-
php-307846-307853	477171	130366	32604	26252	0	0	0	0	0	43891	-	-
php-311346-311348	1030763	123671	6068	6068	38	38	45	45	2	303	1	1
php-307914-307915	539078	175191	55905	49061	1	0	1	0	1	1	1	1
php-309111-309159	513188	178631	56649	49491	8	1	8	1	0	31831	-	-
php-309892-309910	498669	177309	48111	46807	17	17	22	22	3	5300	1	1

Table 19: Prophet-2000 Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	448318	141182	45987	23728	27	24	54	51	0	-	-	-
libtiff-5b0217-3dfb33	1941177	1352092	21932	21903	111	111	2276	2276	0	91329	-	-
libtiff-d13be7-ccadf4	2353240	1771135	34658	34608	103	103	3559	3446	2	2820	1	1
lighttpd-2661-2662	1269307	1016413	115409	103056	57	23	92	58	0	1197010	-	-
lighttpd-1913-1914	1183286	935121	57046	54412	40	40	108	108	0	81194	-	-
python-69934-69935	863452	386935	34011	28504	0	0	0	0	0	-	-	-
gmp-13420-13421	389124	104714	58786	20230	6	3	21	18	1	22027	4	19
gzip-a1d3d4-f17cbd	341681	92972	137153	43377	16	0	16	0	2	4299	1	1
python-70056-70059	526778	236707	5424	5420	0	0	0	0	0	7689	-	-
fbc-5458-5459	16353	5416	6663	4350	34	34	118	89	0	33	-	-
libtiff-ee2ce5-b5691a	3237374	2609366	53567	53119	101	101	1964	1964	1	2601	1	1
php-310991-310999	783770	185001	56048	55384	1	1	1	1	1	8294	1	1
php-308734-308761	440926	151934	43302	38805	0	0	0	0	0	45726	-	-
php-308262-308315	637558	192659	15637	15631	9	9	19	19	1	8304	2	2
php-307562-307561	637588	163053	60720	53760	1	0	1	0	1	34355	1	1
php-309579-309580	621880	198376	33743	33664	11	11	38	38	1	7010	6	11
php-310011-310050	582198	162849	653	653	0	0	0	0	0	13331	-	-
php-309688-309716	609953	191952	7138	7121	13	13	40	40	0	34218	-	-
php-309516-309535	496992	136030	34112	27593	0	0	0	0	0	41978	-	-
php-307846-307853	477171	130366	32634	26252	0	0	0	0	0	43891	-	-
php-311346-311348	1030763	123671	6068	6068	3	3	45	45	1	303	1	1
php-307914-307915	539078	175191	56054	49117	1	0	1	0	1	1	1	1
php-309111-309159	513188	178631	56630	49491	8	1	8	1	0	31831	-	-
php-309892-309910	498669	177309	33432	33262	0	0	0	0	0	5300	-	-

Table 20: Prophet-2000-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	780425	141182	35880	14864	30	28	108	106	0	87034	-	-
libtiff-5b0217-3dfb33	2027218	1352092	71298	51369	154	149	157	152	0	112099	-	-
libtiff-d13be7-ccadf4	2414317	1771135	113033	104624	416	416	431	416	1	4324	1	1
lighttpd-2661-2662	1290186	1016413	110266	95506	60	11	68	19	0	1217473	-	-
lighttpd-1913-1914	1206699	935121	60268	55919	48	48	48	48	0	-	-	-
python-69934-69935	937134	386935	34097	28463	0	0	0	0	0	55996	-	-
gmp-13420-13421	823835	104714	61757	17041	6	5	6	5	1	48675	6	6
gzip-a1d3d4-f17cbd	399573	92972	127371	34577	14	0	14	0	1	6336	2	2
python-70056-70059	573639	236707	5114	5110	0	0	0	0	0	-	-	-
fbc-5458-5459	18782	5416	9052	4762	36	34	56	43	2	24	1	1
libtiff-ee2ce5-b5691a	3311995	2609366	126968	113041	328	328	328	328	1	4478	1	1
php-310991-310999	802073	185001	45417	44536	1	1	1	1	1	19354	1	1
php-308734-308761	464785	151934	43563	38707	0	0	0	0	0	46137	-	-
php-308262-308315	658758	192659	58189	56960	1	1	2	2	1	8635	1	1
php-307562-307561	651312	163053	60650	53423	0	0	0	0	0	38971	-	-
php-309579-309580	629763	198376	64486	59261	2	2	2	2	1	8104	1	1
php-310011-310050	599295	162849	666	666	0	0	0	0	0	16975	-	-
php-309688-309716	621923	191952	47000	46831	31	31	32	32	0	46337	-	-
php-309516-309535	561904	136030	34972	27427	0	0	0	0	0	42385	-	-
php-307846-307853	541823	130366	32871	26033	0	0	0	0	0	45315	-	-
php-311346-311348	1404751	123671	5524	5524	38	38	43	43	2	411	1	1
php-307914-307915	549783	175191	55642	48600	1	0	1	0	1	3	1	1
php-309111-309159	539576	178631	58288	50205	1	1	1	1	0	39224	-	-
php-309892-309910	520768	177309	46811	45988	17	17	22	22	3	15340	1	1

Table 21: Prophet-2000-RExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	780425	141182	39259	15947	27	24	54	51	0	87034	-	-
libtiff-5b0217-3dfb33	2027218	1352092	22673	22656	111	111	2178	2178	0	112099	-	-
libtiff-d13be7-ccadf4	2414317	1771135	37308	37266	93	93	3371	3258	2	4324	1	1
lighttpd-2661-2662	1290186	1016413	104869	90483	69	23	102	56	0	1217473	-	-
lighttpd-1913-1914	1206699	935121	57257	54013	42	42	109	109	0	86441	-	-
python-69934-69935	937134	386935	34092	28463	0	0	0	0	0	55996	-	-
gmp-13420-13421	823835	104714	60747	16845	4	3	19	18	1	48675	4	19
gzip-a1d3d4-f17cbd	399573	92972	127959	34611	14	0	14	0	1	6336	2	2
python-70056-70059	573639	236707	5048	5044	0	0	0	0	0	8292	-	-
fbc-5458-5459	18782	5416	6972	4007	36	34	120	89	0	24	-	-
libtiff-ee2ce5-b5691a	3311995	2609366	48940	48555	90	90	1952	1952	1	4478	1	1
php-310991-310999	802073	185001	45417	44536	1	1	1	1	1	19354	1	1
php-308734-308761	464785	151934	43596	38707	0	0	0	0	0	46137	-	-
php-308262-308315	658758	192659	14190	14158	9	9	21	21	1	8635	2	2
php-307562-307561	651312	163053	60650	53423	0	0	0	0	0	38971	-	-
php-309579-309580	629763	198376	35100	34932	11	11	38	38	1	8104	6	11
php-310011-310050	599295	162849	666	666	0	0	0	0	0	16975	-	-
php-309688-309716	621923	191952	7147	7120	13	13	40	40	0	46337	-	-
php-309516-309535	561904	136030	34838	27396	0	0	0	0	0	42385	-	-
php-307846-307853	541823	130366	32846	26033	0	0	0	0	0	45315	-	-
php-311346-311348	1404751	123671	5524	5524	3	3	42	42	1	411	1	1
php-307914-307915	549783	175191	55601	48600	1	0	1	0	1	3	1	1
php-309111-309159	539576	178631	58271	50205	1	1	1	1	0	39224	-	-
php-309892-309910	520768	177309	11497	11492	11	11	39	39	1	15340	1	1

Table 22: Prophet-2000-RExt-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	10949	3394	10949	3394	33	28	111	106	0	-	-	-
libtiff-5b0217-3dfb33	83937	19298	64840	19298	174	147	177	150	0	-	-	-
libtiff-d13be7-ccadf4	184068	141307	45961	45374	1723	1723	2003	1723	1	2408	3	3
lighttpd-2661-2662	34688	27541	34688	27541	70	8	73	11	0	-	-	-
lighttpd-1913-1914	32903	21707	32903	21707	4	4	4	4	0	-	-	-
python-69934-69935	17828	5874	6960	4981	0	0	0	0	0	-	-	-
gmp-13420-13421	29703	5319	29703	5319	0	0	0	0	0	-	-	-
gzip-a1d3d4-f17cbd	25303	7535	25303	7535	5	0	5	0	1	12083	4	4
python-70056-70059	22537	9353	4254	3971	0	0	0	0	0	-	-	-
fbc-5458-5459	4864	2144	4864	2144	26	26	46	35	2	273	8	11
libtiff-ee2ce5-b5691a	34863	25186	34863	25186	328	328	328	328	1	3692	1	1
php-310991-310999	47688	11762	27894	11762	2	2	2	2	2	1348	1	1
php-308734-308761	14	11	14	11	0	0	0	0	0	-	-	-
php-308262-308315	22003	1869	5933	1869	0	0	0	0	0	2176	-	-
php-307562-307561	21252	4869	11682	4869	1	0	1	0	1	3224	1	1
php-309579-309580	41568	6890	22390	6890	2	2	2	2	1	46	1	1
php-310011-310050	34470	8642	4739	3465	52	6	54	9	0	13476	-	-
php-309688-309716	47600	11980	7073	5111	65	64	67	66	0	5240	-	-
php-309516-309535	19142	4578	19142	4578	1	0	1	0	1	2584	1	1
php-307846-307853	13971	3009	13971	3009	1	0	1	0	1	2570	1	1
php-311346-311348	8096	3027	8096	3027	50	38	72	60	2	465	1	1
php-307914-307915	25707	7731	25707	7731	1	0	1	0	1	3135	1	1
php-309111-309159	32541	6583	22565	6583	10	1	10	1	1	15538	10	10
php-309892-309910	8665	1432	8665	1432	21	17	26	22	4	92	1	1

Table 23: SPR-100 Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	10949	3394	10949	3394	29	24	56	51	0	-	-	-
libtiff-5b0217-3dfb33	83937	19298	13487	5683	162	162	2879	2879	0	-	-	-
libtiff-d13be7-ccadf4	184068	141307	5269	5269	247	247	4450	4450	2	2408	5	104
lighttpd-2661-2662	34688	27541	34688	27541	78	16	109	47	0	-	-	-
lighttpd-1913-1914	32903	21707	32903	21707	4	4	12	12	0	-	-	-
python-69934-69935	17828	5874	6963	4984	0	0	0	0	0	-	-	-
gmp-13420-13421	29703	5319	29703	5319	0	0	0	0	0	-	-	-
gzip-a1d3d4-f17cbd	25303	7535	25303	7535	5	0	5	0	1	12083	4	4
python-70056-70059	22537	9353	4773	4353	0	0	0	0	0	-	-	-
fbc-5458-5459	4864	2144	4864	2144	23	23	98	71	0	273	-	-
libtiff-ee2ce5-b5691a	34863	25186	4828	4185	101	101	3829	3829	1	3692	1	1
php-310991-310999	47688	11762	27444	11762	2	2	2	2	2	1348	1	1
php-308734-308761	14	11	14	11	0	0	0	0	0	-	-	-
php-308262-308315	22003	1869	2416	784	12	12	34	34	1	2176	12	25
php-307562-307561	21252	4869	11831	4869	1	0	1	0	1	3224	1	1
php-309579-309580	41568	6890	15680	6890	11	11	38	38	1	46	1	1
php-310011-310050	34470	8642	4739	3465	52	6	58	13	0	13476	-	-
php-309688-309716	47600	11980	6235	4276	15	14	70	69	0	5240	-	-
php-309516-309535	19142	4578	19142	4578	1	0	1	0	1	2584	1	1
php-307846-307853	13971	3009	13971	3009	1	0	1	0	1	2570	1	1
php-311346-311348	8096	3027	1030	1022	31	30	102	101	2	465	1	1
php-307914-307915	25707	7731	25707	7731	1	0	1	0	1	3135	1	1
php-309111-309159	32541	6583	22886	6583	10	1	10	1	1	15538	10	10
php-309892-309910	8665	1432	353	187	11	11	97	97	1	92	1	1

Table 24: SPR-100-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	12718	3394	12718	3394	33	28	111	106	0	-	-	-
libtiff-5b0217-3dfb33	88613	19298	68340	19298	174	147	177	150	0	-	-	-
libtiff-d13be7-ccadf4	186777	141307	160494	141307	1723	1723	2003	1723	1	2117	3	3
lighttpd-2661-2662	35810	27541	35810	27541	66	4	66	4	0	-	-	-
lighttpd-1913-1914	34917	21707	34917	21707	4	4	4	4	0	-	-	-
python-69934-69935	18168	5874	7329	5320	0	0	0	0	0	-	-	-
gmp-13420-13421	49743	5319	49743	5319	0	0	0	0	0	-	-	-
gzip-a1d3d4-f17cbd	31251	7535	31251	7535	5	0	5	0	1	17958	4	4
python-70056-70059	26398	9353	4794	4365	0	0	0	0	0	-	-	-
fbc-5458-5459	5938	2144	5938	2144	28	26	48	35	2	264	8	11
libtiff-ee2ce5-b5691a	37661	25186	37661	25186	328	328	328	328	1	3680	1	1
php-310991-310999	48843	11762	27311	11762	2	2	2	2	2	1289	1	1
php-308734-308761	14	11	14	11	0	0	0	0	0	-	-	-
php-308262-308315	22472	1869	3304	1211	2	2	4	4	1	2177	2	3
php-307562-307561	21944	4869	11937	4869	1	0	1	0	1	3227	1	1
php-309579-309580	42322	6890	22265	6890	2	2	2	2	1	46	1	1
php-310011-310050	35256	8642	4689	3465	52	6	55	9	0	14262	-	-
php-309688-309716	48118	11980	7083	5111	63	62	64	63	1	5240	34	35
php-309516-309535	21008	4578	21008	4578	1	0	1	0	1	2583	1	1
php-307846-307853	16435	3009	16435	3009	1	0	1	0	1	2616	1	1
php-311346-311348	8173	3027	8173	3027	51	38	73	60	2	455	1	1
php-307914-307915	27121	7731	27121	7731	1	0	1	0	1	3135	1	1
php-309111-309159	34080	6583	23959	6583	10	1	10	1	1	17075	8	8
php-309892-309910	10855	1432	2556	1432	69	17	74	22	3	224	1	1

Table 25: SPR-100-RExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	12718	3394	12718	3394	29	24	56	51	0	-	-	-
libtiff-5b0217-3dfb33	88613	19298	10264	5607	136	136	2721	2721	0	-	-	-
libtiff-d13be7-ccadf4	186777	141307	5269	5269	232	232	4350	4350	2	2117	5	104
lighttpd-2661-2662	35810	27541	35810	27541	74	12	97	35	0	-	-	-
lighttpd-1913-1914	34917	21707	34917	21707	4	4	12	12	0	-	-	-
python-69934-69935	18168	5874	7047	5066	0	0	0	0	0	-	-	-
gmp-13420-13421	49743	5319	49743	5319	0	0	0	0	0	-	-	-
gzip-a1d3d4-f17cbd	31251	7535	31251	7535	5	0	5	0	1	17958	4	4
python-70056-70059	26398	9353	4825	4365	0	0	0	0	0	-	-	-
fbc-5458-5459	5938	2144	5938	2144	25	23	100	71	0	264	-	-
libtiff-ee2ce5-b5691a	37661	25186	4765	4121	101	101	3851	3851	1	3680	1	1
php-310991-310999	48843	11762	27441	11762	2	2	2	2	2	1289	1	1
php-308734-308761	14	11	14	11	0	0	0	0	0	-	-	-
php-308262-308315	22472	1869	2416	784	16	16	39	39	1	2177	12	25
php-307562-307561	21944	4869	11967	4869	1	0	1	0	1	3227	1	1
php-309579-309580	42322	6890	15360	6890	11	11	38	38	1	46	1	1
php-310011-310050	35256	8642	4679	3465	48	6	53	13	0	14262	-	-
php-309688-309716	48118	11980	6235	4276	15	14	68	67	0	5240	-	-
php-309516-309535	21008	4578	21008	4578	1	0	1	0	1	2583	1	1
php-307846-307853	16435	3009	16435	3009	1	0	1	0	1	2616	1	1
php-311346-311348	8173	3027	1040	1022	32	30	105	103	2	455	1	1
php-307914-307915	27121	7731	27121	7731	1	0	1	0	1	3135	1	1
php-309111-309159	34080	6583	23843	6583	10	1	10	1	1	17075	8	8
php-309892-309910	10855	1432	353	187	11	11	102	102	1	224	1	1

Table 26: SPR-100-RExt-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	39287	14161	33154	14161	34	28	112	106	0	-	-	-
libtiff-5b0217-3dfb33	221134	65017	107534	65017	237	149	240	152	1	56644	208	211
libtiff-d13be7-ccadf4	296426	228659	230917	214697	1723	1723	2003	1723	1	372	3	3
lighttpd-2661-2662	120263	98028	105423	98028	52	4	55	7	0	-	-	-
lighttpd-1913-1914	68199	48133	45191	40674	55	55	55	55	0	-	-	-
python-69934-69935	47544	13775	11024	8634	0	0	0	0	0	-	-	-
gmp-13420-13421	50672	8763	50672	8763	3	0	3	0	2	14645	1	1
gzip-a1d3d4-f17cbd	48702	15104	48702	15104	14	0	14	0	1	21926	4	4
python-70056-70059	39599	17961	4126	3652	0	0	0	0	0	-	-	-
fbc-5458-5459	9857	4495	9791	4495	37	37	61	46	2	454	8	11
libtiff-ee2ce5-b5691a	171379	106867	14068	13454	15	15	15	15	1	13296	1	1
php-310991-310999	89230	18988	31084	16653	2	2	2	2	2	384	1	1
php-308734-308761	14692	4160	14692	4160	4	4	4	4	2	5771	1	1
php-308262-308315	90431	10845	9137	8516	0	0	0	0	0	7191	-	-
php-307562-307561	31597	6997	13425	6997	1	0	1	0	1	4918	1	1
php-309579-309580	60351	11416	25400	11416	2	2	2	2	1	46	1	1
php-310011-310050	77671	16558	8160	7316	32	32	49	49	0	30647	-	-
php-309688-309716	71633	15744	10699	6516	31	30	32	31	0	8398	-	-
php-309516-309535	27098	6314	27098	6314	1	0	1	0	1	4000	1	1
php-307846-307853	22131	4757	21654	4757	1	0	1	0	1	3867	1	1
php-311346-311348	9799	3879	9799	3879	50	38	72	60	2	312	1	1
php-307914-307915	47988	15066	43285	15066	1	0	1	0	1	5748	1	1
php-309111-309159	52908	12232	29459	12232	10	1	10	1	1	24347	10	10
php-309892-309910	40758	9999	17455	9999	17	17	22	22	3	179	1	1

Table 27: SPR-200 Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	39287	14161	34474	14161	30	24	57	51	0	-	-	-
libtiff-5b0217-3dfb33	221134	65017	13644	8420	111	111	2604	2604	0	56644	-	-
libtiff-d13be7-ccadf4	296426	228659	13486	13486	225	225	4240	4240	2	372	5	104
lighttpd-2661-2662	120263	98028	105296	98028	36	12	57	33	0	-	-	-
lighttpd-1913-1914	68199	48133	41374	37396	49	49	125	125	0	-	-	-
python-69934-69935	47544	13775	11862	9464	0	0	0	0	0	-	-	-
gmp-13420-13421	50672	8763	50672	8763	3	0	3	0	2	14645	1	1
gzip-a1d3d4-f17cbd	48702	15104	48702	15104	14	0	14	0	1	21926	4	4
python-70056-70059	39599	17961	4364	3836	0	0	0	0	0	-	-	-
fbc-5458-5459	9857	4495	7000	4495	34	34	120	89	0	454	-	-
libtiff-ee2ce5-b5691a	171379	106867	16480	14448	101	101	3603	3603	1	13296	1	1
php-310991-310999	89230	18988	31084	16653	2	2	2	2	2	384	1	1
php-308734-308761	14692	4160	14692	4160	4	4	4	4	2	5771	1	1
php-308262-308315	90431	10845	9167	8516	0	0	0	0	0	7191	-	-
php-307562-307561	31597	6997	13565	6997	1	0	1	0	1	4918	1	1
php-309579-309580	60351	11416	17487	11416	11	11	38	38	1	46	1	1
php-310011-310050	77671	16558	10005	7386	21	20	38	37	0	30647	-	-
php-309688-309716	71633	15744	9861	5681	15	14	38	37	0	8398	-	-
php-309516-309535	27098	6314	27098	6314	1	0	1	0	1	4000	1	1
php-307846-307853	22131	4757	21674	4757	1	0	1	0	1	3867	1	1
php-311346-311348	9799	3879	1329	1319	31	30	104	103	2	312	1	1
php-307914-307915	47988	15066	43355	15066	1	0	1	0	1	5748	1	1
php-309111-309159	52908	12232	29464	12232	10	1	10	1	1	24347	10	10
php-309892-309910	40758	9999	4986	4931	11	11	92	92	1	179	1	1

Table 28: SPR-200-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	61171	14161	50781	14161	33	28	111	106	0	-	-	-
libtiff-5b0217-3dfb33	278857	65017	104524	65017	1164	149	1167	152	1	56646	209	212
libtiff-d13be7-ccadf4	300745	228659	226189	210869	1723	1723	2003	1723	1	6080	3	3
lighttpd-2661-2662	124449	98028	108906	98028	9	3	10	4	0	-	-	-
lighttpd-1913-1914	70712	48133	45056	40544	55	55	55	55	0	-	-	-
python-69934-69935	48560	13775	11024	8634	0	0	0	0	0	24561	-	-
gmp-13420-13421	79763	8763	74674	8763	2	0	2	0	2	40506	1	1
gzip-a1d3d4-f17cbd	58432	15104	58432	15104	14	0	14	0	1	31657	4	4
python-70056-70059	43598	17961	4364	3836	0	0	0	0	0	-	-	-
fbc-5458-5459	11828	4495	10288	4495	39	37	63	46	2	573	8	11
libtiff-ee2ce5-b5691a	190629	106867	167664	106867	328	328	328	328	1	13296	1	1
php-310991-310999	90936	18988	30573	16385	2	2	2	2	2	403	1	1
php-308734-308761	18784	4160	18784	4160	4	4	4	4	2	5728	1	1
php-308262-308315	92459	10845	9197	8516	0	0	0	0	0	7366	-	-
php-307562-307561	32546	6997	13573	6997	1	0	1	0	1	4918	1	1
php-309579-309580	61407	11416	24205	11416	2	2	2	2	1	46	1	1
php-310011-310050	78981	16558	7960	7316	32	32	49	49	0	31956	-	-
php-309688-309716	73424	15744	10699	6516	32	31	33	32	0	8398	-	-
php-309516-309535	29514	6314	29514	6314	1	0	1	0	1	3999	1	1
php-307846-307853	25447	4757	24349	4757	1	0	1	0	1	3868	1	1
php-311346-311348	9978	3879	9568	3879	52	38	74	60	2	312	1	1
php-307914-307915	50442	15066	44514	15066	1	0	1	0	1	5748	1	1
php-309111-309159	58903	12232	32717	12232	10	1	10	1	1	30340	10	10
php-309892-309910	52975	9999	17434	9999	17	17	22	22	3	716	1	1

Table 29: SPR-200-RExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	61171	14161	51587	14161	29	24	56	51	0	-	-	-
libtiff-5b0217-3dfb33	278857	65017	12964	8420	111	111	2604	2604	0	56646	-	-
libtiff-d13be7-ccadf4	300745	228659	13486	13486	232	232	4329	4329	2	6080	5	104
lighttpd-2661-2662	124449	98028	108266	98028	10	4	16	10	0	-	-	-
lighttpd-1913-1914	70712	48133	41374	37396	48	48	123	123	0	-	-	-
python-69934-69935	48560	13775	11024	8634	0	0	0	0	0	24561	-	-
gmp-13420-13421	79763	8763	74666	8763	2	0	2	0	2	40506	1	1
gzip-a1d3d4-f17cbd	58432	15104	58432	15104	14	0	14	0	1	31657	4	4
python-70056-70059	43598	17961	4364	3836	0	0	0	0	0	-	-	-
fbc-5458-5459	11828	4495	6799	4495	34	34	120	89	0	573	-	-
libtiff-ee2ce5-b5691a	190629	106867	16480	14448	101	101	3591	3591	1	13296	1	1
php-310991-310999	90936	18988	30423	16385	2	2	2	2	2	403	1	1
php-308734-308761	18784	4160	18784	4160	4	4	4	4	2	5728	1	1
php-308262-308315	92459	10845	9197	8516	0	0	0	0	0	7366	-	-
php-307562-307561	32546	6997	13518	6997	1	0	1	0	1	4918	1	1
php-309579-309580	61407	11416	14155	8539	11	11	36	36	1	46	1	1
php-310011-310050	78981	16558	9895	7316	21	20	38	37	0	31956	-	-
php-309688-309716	73424	15744	9861	5681	15	14	35	34	0	8398	-	-
php-309516-309535	29514	6314	29514	6314	1	0	1	0	1	3999	1	1
php-307846-307853	25447	4757	24329	4757	1	0	1	0	1	3868	1	1
php-311346-311348	9978	3879	1329	1319	31	30	105	104	2	312	1	1
php-307914-307915	50442	15066	44544	15066	1	0	1	0	1	5748	1	1
php-309111-309159	58903	12232	33002	12232	10	1	10	1	1	30340	10	10
php-309892-309910	52975	9999	4986	4931	11	11	89	89	1	716	1	1

Table 30: SPR-200-RExt-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	59327	22055	41316	22055	34	28	112	106	0	-	-	-
libtiff-5b0217-3dfb33	276142	85251	123778	85251	237	149	240	152	1	59068	206	209
libtiff-d13be7-ccadf4	568172	432083	250826	237142	1723	1723	2003	1723	1	6126	3	3
lighttpd-2661-2662	191579	159862	119941	113050	10	5	13	8	0	-	-	-
lighttpd-1913-1914	159739	114667	37901	34739	32	32	32	32	0	-	-	-
python-69934-69935	65326	18300	11022	8243	0	0	0	0	0	-	-	-
gmp-13420-13421	69661	12794	58934	12794	15	5	15	5	2	20779	6	6
gzip-a1d3d4-f17cbd	82349	18024	82349	18024	14	0	14	0	1	30814	4	4
python-70056-70059	58530	26327	5006	4985	0	0	0	0	0	-	-	-
fbc-5458-5459	16353	5416	9850	5416	37	37	61	46	2	299	5	6
libtiff-ee2ce5-b5691a	218252	142593	203493	142593	328	328	328	328	1	20674	1	1
php-310991-310999	139311	31061	31955	19597	2	2	2	2	2	2256	1	1
php-308734-308761	30623	7437	20855	7437	4	4	4	4	2	7493	1	1
php-308262-308315	150963	19358	14708	14158	0	0	0	0	0	10111	-	-
php-307562-307561	60507	15748	18574	11458	1	0	1	0	1	10019	1	1
php-309579-309580	101940	17784	23578	15481	2	2	2	2	1	46	1	1
php-310011-310050	105542	23824	10543	10543	37	37	55	55	0	41309	-	-
php-309688-309716	95206	19747	13260	7432	7	6	8	7	0	10803	-	-
php-309516-309535	52377	12093	40324	12093	1	0	1	0	1	4695	1	1
php-307846-307853	40272	8051	20723	8051	3	2	4	3	1	5909	1	1
php-311346-311348	14543	5619	8568	5619	52	41	74	63	2	741	4	4
php-307914-307915	64378	20107	45432	20107	1	0	1	0	1	8122	1	1
php-309111-309159	86627	22335	41971	22335	2	1	2	1	0	39752	-	-
php-309892-309910	62347	19484	21735	15396	17	17	22	22	3	641	1	1

Table 31: SPR-300 Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	59327	22055	42448	22055	30	24	57	51	0	-	-	-
libtiff-5b0217-3dfb33	276142	85251	16314	9388	111	111	2598	2598	0	59068	-	-
libtiff-d13be7-ccadf4	568172	432083	19425	19425	203	203	4089	4089	2	6126	5	104
lighttpd-2661-2662	191579	159862	116171	109330	14	9	34	29	0	-	-	-
lighttpd-1913-1914	159739	114667	28702	27455	24	24	53	53	0	107499	-	-
python-69934-69935	65326	18300	11077	8243	0	0	0	0	0	-	-	-
gmp-13420-13421	69661	12794	58745	12794	13	3	28	18	2	20779	4	19
gzip-a1d3d4-f17cbd	82349	18024	82349	18024	14	0	14	0	1	30814	4	4
python-70056-70059	58530	26327	5032	5011	0	0	0	0	0	835	-	-
fbc-5458-5459	16353	5416	6925	4707	34	34	120	89	0	299	-	-
libtiff-ee2ce5-b5691a	218252	142593	24635	22006	101	101	3244	3244	1	20674	1	1
php-310991-310999	139311	31061	32409	19972	2	2	2	2	2	2256	1	1
php-308734-308761	30623	7437	20902	7437	4	4	4	4	2	7493	1	1
php-308262-308315	150963	19358	14748	14158	0	0	0	0	0	10111	-	-
php-307562-307561	60507	15748	18594	11458	1	0	1	0	1	10019	1	1
php-309579-309580	101940	17784	12994	8849	11	11	38	38	1	46	1	1
php-310011-310050	105542	23824	10543	10543	23	23	54	54	0	41309	-	-
php-309688-309716	95206	19747	13260	7432	7	6	8	7	0	10803	-	-
php-309516-309535	52377	12093	41374	12093	1	0	1	0	1	4695	1	1
php-307846-307853	40272	8051	17882	8051	5	4	12	11	1	5909	1	1
php-311346-311348	14543	5619	1655	1644	34	33	97	96	2	741	4	4
php-307914-307915	64378	20107	45791	20107	1	0	1	0	1	8122	1	1
php-309111-309159	86627	22335	42158	22335	2	1	2	1	0	39752	-	-
php-309892-309910	62347	19484	4917	4917	11	11	85	85	1	641	1	1

Table 32: SPR-300-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	95751	22055	54107	22055	39	28	117	106	1	50188	32	110
libtiff-5b0217-3dfb33	336808	85251	123207	85251	1164	149	1165	155	1	59066	209	215
libtiff-d13be7-ccadf4	573714	432083	87760	87073	309	309	309	309	1	4629	3	3
lighttpd-2661-2662	197676	159862	120127	113198	8	3	8	3	0	-	-	-
lighttpd-1913-1914	168946	114667	37891	34739	32	32	32	32	0	-	-	-
python-69934-69935	66616	18300	12534	9511	0	0	0	0	0	32709	-	-
gmp-13420-13421	107650	12794	78103	12794	41	5	41	5	1	46636	6	6
gzip-a1d3d4-f17cbd	100652	18024	91433	18024	14	0	14	0	1	49115	4	4
python-70056-70059	64192	26327	5032	5011	0	0	0	0	0	-	-	-
fbc-5458-5459	18782	5416	12524	5416	34	32	55	38	2	582	8	11
libtiff-ee2ce5-b5691a	241707	142593	195947	142593	328	328	328	328	1	20686	1	1
php-310991-310999	142164	31061	31855	19597	2	2	2	2	2	2139	1	1
php-308734-308761	36012	7437	21042	7437	4	4	4	4	2	7493	1	1
php-308262-308315	159696	19358	14748	14158	0	0	0	0	0	10106	-	-
php-307562-307561	64392	15748	18564	11458	1	0	1	0	1	10022	1	1
php-309579-309580	103151	17784	23738	15481	2	2	2	2	1	46	1	1
php-310011-310050	107479	23824	10543	10543	37	37	55	55	0	43288	-	-
php-309688-309716	97372	19747	13259	7432	10	9	11	10	0	10803	-	-
php-309516-309535	56549	12093	43485	12093	1	0	1	0	1	4694	1	1
php-307846-307853	44992	8051	21298	8051	3	2	4	3	1	5909	1	1
php-311346-311348	14889	5619	8607	5619	52	41	74	63	2	741	4	4
php-307914-307915	67286	20107	46603	20107	1	0	1	0	1	8122	1	1
php-309111-309159	102384	22335	43158	22335	2	1	2	1	0	55064	-	-
php-309892-309910	74780	19484	21370	15253	17	17	22	22	3	3305	1	1

Table 33: SPR-300-RExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	95751	22055	54176	22055	35	24	62	51	1	50188	28	55
libtiff-5b0217-3dfb33	336808	85251	13784	9388	111	111	2604	2604	0	59066	-	-
libtiff-d13be7-ccadf4	573714	432083	19425	19425	215	215	4125	4125	2	4629	5	104
lighttpd-2661-2662	197676	159862	116171	109330	16	11	31	26	0	-	-	-
lighttpd-1913-1914	168946	114667	28116	26867	24	24	53	53	0	107498	-	-
python-69934-69935	66616	18300	12773	9592	0	0	0	0	0	32709	-	-
gmp-13420-13421	107650	12794	77443	12794	39	3	54	18	1	46636	4	19
gzip-a1d3d4-f17cbd	100652	18024	91870	18024	14	0	14	0	1	49115	4	4
python-70056-70059	64192	26327	5032	5011	0	0	0	0	0	3165	-	-
fbc-5458-5459	18782	5416	9253	5416	29	27	100	67	0	582	-	-
libtiff-ee2ce5-b5691a	241707	142593	24635	22006	101	101	3412	3412	1	20686	1	1
php-310991-310999	142164	31061	31955	19597	2	2	2	2	2	2139	1	1
php-308734-308761	36012	7437	21072	7437	4	4	4	4	2	7493	1	1
php-308262-308315	159696	19358	14748	14158	0	0	0	0	0	10106	-	-
php-307562-307561	64392	15748	18534	11458	1	0	1	0	1	10022	1	1
php-309579-309580	103151	17784	13274	8849	11	11	38	38	1	46	1	1
php-310011-310050	107479	23824	10543	10543	23	23	54	54	0	43288	-	-
php-309688-309716	97372	19747	12494	6667	7	7	13	13	0	10803	-	-
php-309516-309535	56549	12093	43728	12093	1	0	1	0	1	4694	1	1
php-307846-307853	44992	8051	20127	8051	5	4	12	11	1	5909	1	1
php-311346-311348	14889	5619	1655	1644	34	33	98	97	2	741	4	4
php-307914-307915	67286	20107	46743	20107	1	0	1	0	1	8122	1	1
php-309111-309159	102384	22335	41638	22335	2	1	2	1	0	55064	-	-
php-309892-309910	74780	19484	4917	4917	11	11	87	87	1	3305	1	1

Table 34: SPR-300-RExt-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	448318	141182	48639	40229	28	28	106	106	0	-	-	-
libtiff-5b0217-3dfb33	1941177	1352092	376088	346011	59	59	59	59	0	309678	-	-
libtiff-d13be7-ccadf4	2353240	1771135	537943	510074	39	39	39	39	1	34935	3	3
lighttpd-2661-2662	1269307	1016413	158849	158849	84	84	108	108	1	73333	10	18
lighttpd-1913-1914	1183286	935121	67947	67947	30	30	30	30	0	-	-	-
python-69934-69935	863452	386935	74931	67619	0	0	0	0	0	-	-	-
gmp-13420-13421	389124	104714	77939	51103	5	5	5	5	0	148326	-	-
gzip-a1d3d4-f17cbd	341681	92972	172669	92972	15	0	15	0	1	21929	1	1
python-70056-70059	526778	236707	5549	5549	0	0	0	0	0	-	-	-
fbc-5458-5459	16353	5416	10804	5416	37	37	61	46	2	299	5	6
libtiff-ee2ce5-b5691a	3237374	2609366	559844	545779	328	328	328	328	1	510769	1	1
php-310991-310999	783770	185001	71978	71969	1	1	1	1	1	20811	1	1
php-308734-308761	440926	151934	70790	69209	0	0	0	0	0	70482	-	-
php-308262-308315	637558	192659	56294	56291	0	0	0	0	0	84098	-	-
php-307562-307561	637588	163053	85244	84684	0	0	0	0	0	85214	-	-
php-309579-309580	621880	198376	103822	99184	2	2	2	2	1	28250	1	1
php-310011-310050	582198	162849	4293	4293	0	0	0	0	0	249855	-	-
php-309688-309716	609953	191952	100081	94908	1	0	1	0	0	84053	-	-
php-309516-309535	496992	136030	56498	47458	0	0	0	0	0	58080	-	-
php-307846-307853	477171	130366	58099	49977	0	0	0	0	0	59502	-	-
php-311346-311348	1030763	123671	11552	11552	29	29	47	47	2	8838	1	1
php-307914-307915	539078	175191	89793	83690	1	0	1	0	1	57702	1	1
php-309111-309159	513188	178631	90328	84540	1	1	1	1	0	249184	-	-
php-309892-309910	498669	177309	90902	86247	5	5	5	5	1	4347	1	1

Table 35: SPR-2000 Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	448318	141182	29653	24692	24	24	51	51	0	-	-	-
libtiff-5b0217-3dfb33	1941177	1352092	13873	13873	111	111	2442	2442	0	309678	-	-
libtiff-d13be7-ccadf4	2353240	1771135	288244	284634	52	52	2708	2708	2	34935	5	104
lighttpd-2661-2662	1269307	1016413	141022	141022	117	117	189	189	1	73333	9	14
lighttpd-1913-1914	1183286	935121	60084	60084	37	37	84	84	0	297232	-	-
python-69934-69935	863452	386935	75020	67619	0	0	0	0	0	-	-	-
gmp-13420-13421	389124	104714	69769	46070	3	3	18	18	0	148326	-	-
gzip-a1d3d4-f17cbd	341681	92972	173757	92972	15	0	15	0	1	21929	1	1
python-70056-70059	526778	236707	6816	6816	0	0	0	0	0	11955	-	-
fbc-5458-5459	16353	5416	6746	4640	34	34	120	89	0	299	-	-
libtiff-ee2ce5-b5691a	3237374	2609366	525373	513075	71	71	496	496	1	510769	1	1
php-310991-310999	783770	185001	71978	71969	1	1	1	1	1	20811	1	1
php-308734-308761	440926	151934	70790	69209	0	0	0	0	0	70482	-	-
php-308262-308315	637558	192659	56286	56283	0	0	0	0	0	84098	-	-
php-307562-307561	637588	163053	92960	84711	0	0	0	0	0	85214	-	-
php-309579-309580	621880	198376	81472	81463	11	11	34	34	1	28250	1	1
php-310011-310050	582198	162849	4293	4293	0	0	0	0	0	249855	-	-
php-309688-309716	609953	191952	99991	94908	1	0	1	0	0	84053	-	-
php-309516-309535	496992	136030	56588	47458	0	0	0	0	0	58080	-	-
php-307846-307853	477171	130366	58149	49977	0	0	0	0	0	59502	-	-
php-311346-311348	1030763	123671	11552	11552	28	28	46	46	2	8838	1	1
php-307914-307915	539078	175191	89763	83690	1	0	1	0	1	57702	1	1
php-309111-309159	513188	178631	90378	84540	1	1	1	1	0	249184	-	-
php-309892-309910	498669	177309	3691	3691	11	11	42	42	1	4347	1	1

Table 36: SPR-2000-CExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	780425	141182	27515	23574	10	10	29	29	0	214233	-	-
libtiff-5b0217-3dfb33	2027218	1352092	353982	324268	59	59	59	59	0	309679	-	-
libtiff-d13be7-ccadf4	2414317	1771135	532541	506165	39	39	39	39	1	33699	3	3
lighttpd-2661-2662	1290186	1016413	157305	157305	84	84	105	105	1	73333	10	18
lighttpd-1913-1914	1206699	935121	67947	67947	31	31	31	31	0	-	-	-
python-69934-69935	937134	386935	75716	67619	0	0	0	0	0	470550	-	-
gmp-13420-13421	823835	104714	69833	46134	5	5	5	5	0	174177	-	-
gzip-a1d3d4-f17cbd	399573	92972	183189	92972	5	0	5	0	1	21936	1	1
python-70056-70059	573639	236707	5761	5761	0	0	0	0	0	-	-	-
fbc-5458-5459	18782	5416	14112	5416	34	32	55	38	2	582	8	11
libtiff-ee2ce5-b5691a	3311995	2609366	544755	531755	329	329	329	329	1	510781	1	1
php-310991-310999	802073	185001	71978	71969	1	1	1	1	1	13860	1	1
php-308734-308761	464785	151934	70780	69209	0	0	0	0	0	70482	-	-
php-308262-308315	658758	192659	54186	54183	0	0	0	0	0	84097	-	-
php-307562-307561	651312	163053	92759	84711	0	0	0	0	0	85216	-	-
php-309579-309580	629763	198376	103872	99184	2	2	2	2	1	28294	1	1
php-310011-310050	599295	162849	5151	5151	1	1	2	2	0	266951	-	-
php-309688-309716	621923	191952	99781	94908	1	0	1	0	0	83933	-	-
php-309516-309535	561904	136030	56528	47458	0	0	0	0	0	58080	-	-
php-307846-307853	541823	130366	58119	49977	0	0	0	0	0	59503	-	-
php-311346-311348	1404751	123671	6977	6977	28	28	46	46	2	3218	1	1
php-307914-307915	549783	175191	89743	83690	1	0	1	0	1	57702	1	1
php-309111-309159	539576	178631	90328	84540	1	1	1	1	0	275198	-	-
php-309892-309910	520768	177309	90793	86247	5	5	5	5	1	5439	1	1

Table 37: SPR-2000-RExt Statistics

Defect	Search Space Templates		Evaluated Templates		Plausible Templates		Plausible Patches		Correct Patches	Correct Template Rank		Correct Patch Rank in Plausible
	All	Cond.	All	Cond.	All	Cond.	All	Cond.		In Space	In Plausible	
gmp-14166-14167	780425	141182	27505	23574	6	6	13	13	0	214233	-	-
libtiff-5b0217-3dfb33	2027218	1352092	37909	37909	111	111	2264	2264	0	309679	-	-
libtiff-d13be7-ccadf4	2414317	1771135	285546	283944	52	52	2708	2708	2	33699	5	104
lighttpd-2661-2662	1290186	1016413	136816	136816	117	117	189	189	1	73333	9	14
lighttpd-1913-1914	1206699	935121	64918	64918	28	28	65	65	0	297229	-	-
python-69934-69935	937134	386935	75590	67619	0	0	0	0	0	470550	-	-
gmp-13420-13421	823835	104714	69833	46134	3	3	18	18	0	174177	-	-
gzip-a1d3d4-f17cbd	399573	92972	183149	92972	5	0	5	0	1	21936	1	1
python-70056-70059	573639	236707	5825	5825	0	0	0	0	0	6175	-	-
fbc-5458-5459	18782	5416	8812	5416	27	27	98	67	0	582	-	-
libtiff-ee2ce5-b5691a	3311995	2609366	525373	513075	71	71	471	471	1	510781	1	1
php-310991-310999	802073	185001	71688	71679	1	1	1	1	1	13860	1	1
php-308734-308761	464785	151934	70770	69209	0	0	0	0	0	70482	-	-
php-308262-308315	658758	192659	54186	54183	0	0	0	0	0	84097	-	-
php-307562-307561	651312	163053	85204	84684	0	0	0	0	0	85216	-	-
php-309579-309580	629763	198376	54142	54139	11	11	37	37	1	28294	1	1
php-310011-310050	599295	162849	5151	5151	0	0	0	0	0	266951	-	-
php-309688-309716	621923	191952	99781	94908	1	0	1	0	0	83933	-	-
php-309516-309535	561904	136030	56548	47458	0	0	0	0	0	58080	-	-
php-307846-307853	541823	130366	58119	49977	0	0	0	0	0	59503	-	-
php-311346-311348	1404751	123671	6977	6977	28	28	45	45	2	3218	1	1
php-307914-307915	549783	175191	89723	83690	1	0	1	0	1	57702	1	1
php-309111-309159	539576	178631	90328	84540	1	1	1	1	0	275198	-	-
php-309892-309910	520768	177309	8224	8224	11	11	41	41	1	5439	1	1

Table 38: SPR-2000-RExt-CExt Statistics