A Classification Problem in the Credit Card Industry: A Comparison of Three Solutions

Christopher H. Jolly
GE Corporate Research and Development, Schenectady, NY 12301, USA

Mukkai S. Krishnamoorthy
Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Abstract
Classifying accounts is a critical part of efficiently servicing credit cards. A problem we call the Rule Coverage Problem (RCP) arises when using rules for specifying a classification strategy, and we develop and compare three approaches for solving this problem.

1 Introduction and RCP

GE Capital Services (GECS) is the world's largest supplier and manager of private label credit cards, serving 60 million accounts for 300 retailers. With such a large number of accounts it is critical to perform all servicing functions effectively and efficiently since small improvements on a per-account basis have a significant impact on profits. Two major servicing functions are collections and marketing.

Collecting from delinquent accounts consists of creating and implementing a collection strategy that allocates resources (e.g., collector phone calls and computer-generated letters) to delinquent accounts. For example, "good" customers should be separated from "bad" customers because we collect efficiently from them in different ways. An effective collection strategy maximizes payments less expenses over the long term while maximizing customer good will. A collection strategy is created either by intuition and experience or more recently by optimization programs such as PAYMENT [Makuch90].

Marketing active accounts consists of creating and implementing a marketing strategy that augment billing statements with promotional messages and inserts. For example, credit card insurance only needs to be marketed to those card holders who do not already have it, and a new store opening only needs to be marketed to those living (or working or shopping) within a reasonable driving distance. An effective marketing strategy increases credit card usage, lowers advertising costs, and increases (credit card and retailer) brand loyalty.

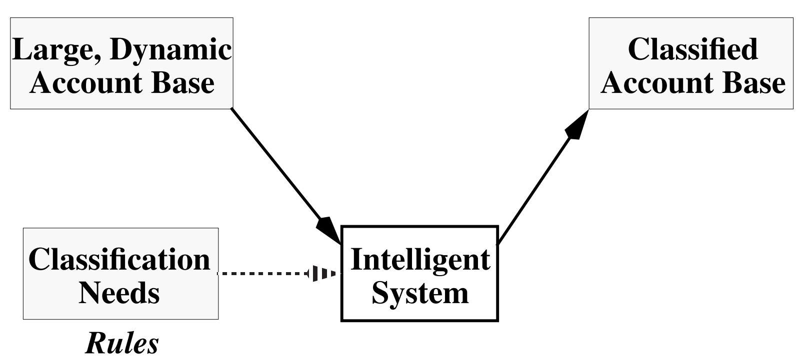

Figure 1: Classification Problem

Thus, GECS needs to be able to classify accounts into homogeneous groups as shown in Figure 1.

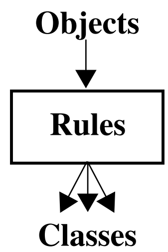

Figure 2: Using Rules For Classification

An efficient method for specifying a classification strategy is with rules as shown in Figure 2. Each rule tests attributes of an object in the rule's antecedent and concludes a class in its consequent (rules have the form "if *antecedent* then *consequent*"). A collection strategy typically consists of hundreds of rules, and a marketing strategy typically consists of tens of rules.

```
if x < 50 then class AA
if y < 10 then class BB
```

Figure 3: Ambiguous for $x = 40$ and $y = 4$

Rules are *ambiguous* when more than one applies. For example, if $x = 40$ and $y = 4$ for the rules in Figure 3, the rules could assign either class *AA* or *BB* since both rules apply. To resolve the ambiguity, we assign priorities to the rules. The rules are written (from top to bottom) from highest priority to lowest priority. Therefore, in this example the rules would assign class *AA*.

1.1 Rule Coverage

Higher priority rules can *cover* (i.e., render useless) lower priority rules. In Figure 4, for example, whenever the antecedent of the second rule is true, the antecedent of the first rule is true. Since the first rule has higher priority, the second rule never will assign class *BB* to any object.

```
if x < 500 then class AA
if x < 400 then class BB
```

Figure 4: Simple Example of Rule Coverage

Under the assumption that a person is specifying classification rules, then a rule coverage is a logical error similar to an unreachable statement in a programming language (see Figure 5). An intelligent rule editor should point out each rule covered and the rules covering it.

```
      if 1 = 1 then goto label AA
      x = 400
AA:   continue
```

Figure 5: Example of an Unreachable Program Statement

Informally, the Rule Coverage Problem (RCP) is determining if any rules are covered by higher-priority rules. It can be shown that RCP is Co-NP-complete (see [Garey79] for an introduction to NP-complete problems).

1.2 Solving RCP

We look for potential algorithms for solving RCP by reducing RCP to a better known problems and examining the algorithms developed for them. The reduction, however, should not be too complicated; otherwise, the algorithms that work well for the better known problem may not work well for RCP after the reduction. In this work we reduce instances of RCP to instances of Co-SAT, which can be solved by many methods [Hooker88]. We solved them using two methods: integer programming and resolution.

Since RCP is Co-NP-complete and some practical classification strategies have over 1000 rules, for our third approach we define a class of subproblems of RCP and develop efficient algorithms for some of them.

All three approaches were tested on sample cases, and the results are shown in Section 4.

2 Reducing RCP to a Co-SAT

It can be shown that RCP reduces to the unsatisfiability problem, Co-SAT, as shown in Figure 6. We reduce RCP to the Single Rule Coverage Problem (SRCP), SRCP to what we define as a theorem proving problem over multiple-valued variables (MVTP), MVTP to what we define as Multi-SAT, and Multi-SAT to SAT. Only the problem definitions are presented here; the reductions and proofs of correctness have been omitted to save space.

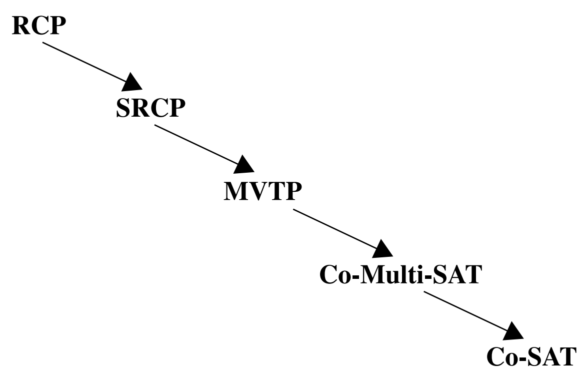

Figure 6: Reductions to a Known Problem

2.1 The Rule Coverage Problem (RCP)

In this section we give the formal definition of the Rule Coverage Problem (RCP). We start with some definitions. Throughout, we assume *n* rules and *m* variables.

Let $V = \{v_1, v_2, \ldots, v_m\}$ be a set of *variables*. Let $T = \{t_{v_1}, t_{v_2}, \ldots, t_{v_m}\}$ be a set of *test* sets. Each t_{v_i} is a set of *simple* boolean tests on variable v_i. Simple boolean tests are not formally defined here, but some examples are $x < 500$, $x = 7$, and $3 < x < 20$.

if x < 75 and y < 5 and z = 0	then class AA
if x < 50 and z = 1	then class BB
if x > 35	then class CC
if y < 2	then class DD
if y > 6 and z = 1	then class EE

Figure 7: Example Classification Rules

For example, for the rules in Figure 7, the variables $V = \{x, y, z\}$, and the set of test sets $T = \{t_x, t_y, t_z\}$, where test set $t_x = \{x < 50, x < 75, x > 35\}$, test set $t_y = \{y < 2, y < 5, y > 6\}$, and test set $t_z = \{z = 0, z = 1\}$.

Let $R = \{r_1, r_2, \ldots, r_n\}$ be a sequence (order is important) of *rules*. A rule typically has two parts: an *antecedent*, the condition under which a rule is true, and a *consequent*, the actions of a rule. Since only the antecedent is relevant here, only the antecedent is shown. A rule is defined here as a conjunction of simple boolean tests. In addition, we often write rules in a more compact form as is done with boolean expressions. We put each test inside parentheses and have implied *and*'s between tests. Figure 8 shows the rules in Figure 7 in shorthand notation.

R1: $(x < 75)\ (y < 5)\ (z = 0)$
R2: $(x < 50)\ (z = 1)$
R3: $(x > 35)$
R4: $(y < 2)$
R5: $(y > 6)\ (z = 1)$

Figure 8: Example Rules

The Rule Coverage Problem RCP(*V*,*T*,*R*) is given variables *V*, tests *T*, and rules *R*, is there a rule r_i that is not satisfiable whenever rules r_j for $j < i$ are false (or, equivalently, is there a rule r_i such that whenever r_i is true, rule r_j, for some $j < i$, is true.)

For example, consider the rules shown in Figure 8. Variable z takes on the values 0 and 1 only. In this case rule *R4* is covered. When *R4* is true, either $x \le 35$ or $x > 35$. If $x > 35$, then *R3* is true. If $x \le 35$, then either $z = 1$ or $z = 0$. If $z = 1$, then *R2* is true. If $z = 0$, then *R1* is true ($y < 5$ since *R4* is true). Thus, rule *R4* is covered.

2.2 The Single Rule Coverage Problem (SRCP)

A natural subproblem of RCP is the Single Rule Coverage Problem (SRCP). The Single Rule Coverage Problem SRCP(*V*,*T*,*R*) is given variable *V*, tests *T*, and rules *R*, is the last rule, r_n, not satisfiable whenever rules r_j, for $j < |R|$, are false (or, equivalently, whenever r_n is true, is there a rule r_j, for some $j < |R|$, that is true).

In first order logic SRCP(*V*,*T*,*R*) is

$$\forall A\Big(\Big(\forall j < n, \overline{r_j(A)}\Big) \Rightarrow \overline{r_n(A)}\Big)$$

or, equivalently,

$$\forall A\ (r_n(A) \Rightarrow \exists j < n, r_j(A))\ ,$$

where *A* represents the variable assignments, and $r_i(A)$ is the boolean value of the rule under assignment *A*.

2.3 Multiple-valued variables

Whereas boolean variables take on the values *true* and *false* (or 1 and 0), we write *multiple-valued variables* to indicate variables that take on integer values from 0 to some finite number. We also refer to such variables as *multi-valued* variables or *multi-variables* for short.

We also define some notation to simplify the use of multi-valued variables. For boolean literals, a variable with a bar over it (such as $\bar{x}$) is true if the variable has the value 0, and a variable without a bar over it (such as x) is true if the variable has the value 1. For multi-valued literals, we list as subscripts the values that make the literal true. For example, if x is a multi-valued variable, the literal x_o is true when *x* has the value 0; x_1 is true when *x* has the value 1; and $x_{0,\,1}$ is true when x has the value 0 or 1. When not ambiguous, we omit the subscripted commas and write $x_{0,\,1}$ as x_{01}. We often write *multi-literal* to mean a literal of a multi-variable.

If a multi-literal has all possible subscripts, then we say it *reduces to true*, since the literal is always satisfied.

2.4 Multi-Variable Theorem Proving Problem (MVTP)

A theorem proving problem using propositional logic is determining if $P \Rightarrow Q$ is a tautology, where *P* and *Q* are expressions over boolean variables. We define the Multi-Variable Theorem Proving Problem (MVTP) as a theorem proving problem over multi-valued variables. Formally, MVTP(*V*,*L*,*P*,*Q*) is given a set of multi-variables $V = \{v_1, v_2, \ldots, v_m\}$ and lengths $L = \{l_{v_1}, l_{v_2}, \ldots, l_{v_m}\}$, where v_i can take on integer values between 0 and $l_{v_i} - 1$, and two expressions, *P* and *Q*, over *V*, is $P \Rightarrow Q$ a tautology.

For example, consider $V = \{w, x, y, z\}$, $P = x_{234}(x_{04} + y_0)w_0$, and $Q = x_{34} + y_{01} + z_0$, where $l_w = 2$, $l_x = 5$, $l_y = 3$, and $l_z = 2$. $P \Rightarrow Q$ is

$$x_{234}(x_{04} + y_0)w_0 \Rightarrow x_{34} + y_{01} + z_0,$$

which is a tautology that can be proved using algebra or a truth table on all possible values for the variables.

2.5 Multi-SAT

Multi-SAT, defined here, is similar to SAT except that the variables are multi-valued variables, not boolean variables. Formally, Multi-SAT(*V*,*L*,*C*) is given a set of multi-variables $V = \{v_1, v_2, \ldots, v_m\}$ and lengths $L = \{l_{v_1}, l_{v_2}, \ldots, l_{v_m}\}$, where v_i can take on integer values between 0 and $l_{v_i} - 1$, and given a set of clauses *C* over the multi-literals from the variables in *V*, is the conjunction of the clauses in *C* satisfiable.

For example, consider $V = \{w, x, y, z\}$ and $C = x_{234}, x_{04} + y_0, w_0, x_{012}, y_2, z_1$, where $l_w = 2$, $l_x = 5$, $l_y = 3$, and $l_z = 2$. The clauses in *C* are not all satisfiable, which can be shown by algebra or a truth table.

3 RCP and incomplete subproblems

Since RCP is Co-NP-complete, we may want to define and solve a tractable subproblem. One natural class of subproblems is *N-rule coverage*. An algorithm performs N-rule coverage if it detects all rules that are covered by N or fewer rules, and it may or may not detect rules that are covered by more than N rules. An example of 1-rule coverage is shown in Figure 4, and an example of 3-rule coverage is shown in Figure 8.

3.1 Strategy for detecting N-rule coverage

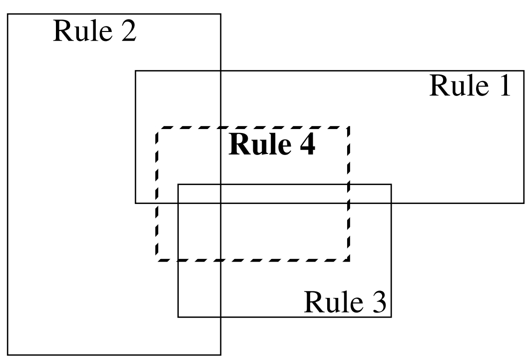

Figure 9: Graphical Representation of Three-Rule Coverage

Rule coverage can be viewed graphically as shown in Figure 9, where rules 1, 2, and 3 cover rule 4. One way of deciding RCP is to determine if each rule is occupying space not covered by higher priority rules.

If for the moment we restrict the variables in RCP to having two values, then we can view RCP as a boolean sum. Consider, for example, the rules in Figure 10. We can represent $(x < 50)$ by $\bar{x}$, $(x \geq 50)$ by x, $(y < 2)$ by $\bar{y}$, and so on

(z only takes on the values 0 and 1), and we obtain the boolean expressions shown in Figure 11. We can record the

R1: $(x < 50)\ (y < 2)\ (z = 0)$

R2: $(x < 50)\ (z = 1)$

R3: $(x \geq 50)$

R4: $(y < 2)$

Figure 10: Rules for Boolean Sum

R1: $\bar{x}\bar{y}\bar{z}$

R2: $\bar{x}z$

R3: x

R4: $\bar{y}$

Figure 11: Expressions for Mathematical Sum

space occupied by keeping track of the sum. Beginning with the highest priority rule, the space occupied by *R1* is $\bar{x}\bar{y}\bar{z}$. We sequentially add the next highest priority rule and check if the total space occupied by the sum increases. Adding *R2*, the sum, $\bar{x}\bar{y}\bar{z} + \bar{x}z$, simplifies to $\bar{x}\bar{y} + \bar{x}z$, which occupies more space than $\bar{x}\bar{y}\bar{z}$ because the former includes $\bar{x}yz$. Adding *R3*, the sum, $\bar{x}\bar{y} + \bar{x}z + x$, simplifies to $\bar{y} + z + x$, which occupies more space than the preceding sum because it includes xyz. Adding *R4*, the sum, $\bar{y} + z + x + x$, simplifies to $\bar{y} + z + x$. Since this sum is identical to the previous sum, no unoccupied space was added to the sum, and thus rule *R4* is covered.

One flaw in this procedure is that minimal boolean sums are not unique, and thus determining if adding a term to a sum increases the total occupied space is a difficult problem (as difficult as SRCP). To overcome this flaw we can represent the sum in a canonical form such as all prime implicants, which can be computed by the consensus procedure of Quine [Schneeweiss89]. Another flaw of this summing procedure is that we restricted it to boolean variables. We overcome this flaw by extending Quine's consensus procedure for use with multi-variables.

3.2 Multi-variable consensus procedure

The consensus procedure of Quine can be extended for use with multi-variables to form a multi-variable consensus procedure. We also add a continuation rule, which like the simplification rule speeds up the procedure when implemented on a computer. The multi-variable consensus procedure is useful in solving RCP because we can compute all prime implicants in a running "sum", and if a rule is absorbed by any multi-variable prime implicant, it is covered.

Before presenting the multi-variable consensus rules, we introduce some notation. The superscripted minus on a variable or a term T (T^-) indicates that the range of each multi-variable in T^- is a (non-strict) non-empty subset of the range of the corresponding variable in *T*. For example, if $T = x_{12}y_{04}$, then T^- might be $x_{12}y_{04}$, x_1y_{04}, x_2y_0, etc. Similarly the superscripted plus on a variable or a term T (T^+) indicates that the range of each multi-variable in T^+ is a (non-strict) superset of the range of the corresponding variable in *T*.

3.2.1 The multi-variable absorption rule

The multi-variable absorption rule is

$$T + T^- U = T,$$

where *T* and *U* are multi-variable terms. (The absorption rule is $T + TU = T$.) For example, $w_{024}x_{12} + w_{04}x_{12}y_{23}z_{01}$ becomes $w_{024}x_{12}$.

3.2.2 The multi-variable continuation rule

The multi-variable continuation rule is

$$x_r T + x_s T = x_{r \cup s} T,$$

where x is a multi-variable and T is a multi-variable term. For example, $x_{024}y_{12} + x_{14}y_{12}$ becomes $x_{0124}y_{12}$. Multi-variable $x_{r \cup s}$ is replaced by *true* if $r \cup s$ includes all ranges. The multi-variable continuation rule is not applied if $r \cup s = r$ or $r \cup s = s$. Note that the multi-variable continuation rule is derived from the multi-variable simplification rule when $T = T^{-}$ and U is empty, followed by the multi-variable absorption rule.

3.2.3 The multi-variable simplification rule

The multi-variable simplification rule is

$$x_r T + x_s T^{-} U = x_r T + x_{r \cup s} T^{-} U,$$

where x is a multi-variable and T and U are multi-variable terms. (The simplification rule is $\underset{\sim}{x} + \bar{\underset{\sim}{x}}T = \underset{\sim}{x} + T$.) For example, $x_{024}y_{12} + x_{14}y_2z_{01}$ becomes $x_{024}y_{12} + x_{0124}y_2z_{01}$. Multi-variable $x_{r \cup s}$ is replaced by *true* if $r \cup s$ includes all ranges. The multi-variable simplification rule is not applied if $r \cup s = s$. (Note that Quine's simplification rule could be written more generally as $\underset{\sim}{x}T + \bar{\underset{\sim}{x}}TU = \underset{\sim}{x}T + TU$, which is derived from one application of the consensus rule, followed by one application of the absorption rule.)

3.2.4 The multi-variable consensus rule

The multi-variable consensus rule is to add a multi-variable consensus of two multi-variable terms. The multi-variable consensus of $x_r T$ and $x_s U$ is

$$\mathrm{Q}(x_r T, x_s U; \mathrm{x}) = x_{r \cup s} TU.$$

(The consensus of xT and $\bar{x}U$ is $\mathrm{Q}(xT, \bar{x}U; \mathrm{x}) = TU$.) For example, $x_{024}y_{12}w_1 + x_{14}y_2z_{01}$ becomes $x_{024}y_{12}w_1 + x_{14}y_2z_{01} + x_{0124}y_2z_{01}w_1$. Multi-variable $x_{r \cup s}$ is replaced by *true* if $r \cup s$ includes all ranges. The consensus rule is not applied if $r \cup s = r$, $r \cup s = s$, or $TU = \phi$.

3.3 Rule coverage

The multi-variable consensus rules can be used to create 1-rule, 2-rule, and 3-rule coverage algorithms using the following additional rule, which we call the partial absorption rule.

$$x_r T + x_s T^{-} U = x_r T + x_{s-r} T^{-} U,$$

where $x_{s-r}T^{-}U$ is omitted if $s \subset r$. Note that the partial absorption rule is a more general form of the multi-variable absorption rule. When $s \subset r$, $x_r T$ absorbs $x_s T^{-} U$. Only the part of $x_s T^{-} U$ that is absorbed is removed.

Theorem 1: When a rule, *R,* is covered under 2-rule coverage, either (1) applying partial absorption on *R*, once for each of the two rules, absorbs *R*, or (2) *R* is only absorbed by a result of applying a multi-variable rule on the two covering rules where a variable is reduced to *true*.

We illustrate the two cases with simple examples. Assume terms T_1 and T_2 cover T_a. If $T_1 = x_{024}y_{12}w_1$ and $T_2 = x_{14}y_2z_{01}$, then $T_a = x_{01}y_2z_0w_1$ is absorbed after applying the partial absorption rule twice, first with T_1 to reduce T_a to $x_1y_2z_0w_1$, and then with T_2, which absorbs it. If $T_1 = x_{024}y_{12}w_1$ and $T_2 = x_{134}y_2z_{01}$, where $x_{01234} = \mathit{true}$, then $T_a = y_2z_0w_1$ is absorbed by the consensus of T_1 and T_2, which is y_2z_{01}.

Theorem 2: When a rule, *R,* is covered under 3-rule coverage, either (1) applying partial absorption on *R* twice, once for each of the three rules and then again on the three rules, absorbs *R*, or (2) *R* is only absorbed by a result of applying the multi-variable rules on the three covering rules where a variable is reduced to *true*.

We illustrate with an example. Assume terms T_1, T_2, and T_3 cover T_a. If $T_1 = x_0y_1$, $T_2 = x_1y_1$, and $T_3 = x_{01}y_0$, then $T_a = x_{01}y_{01}$ is absorbed after applying the partial absorption rule in two passes. On the first pass T_a is not altered by T_1 or T_2 but is by T_3 to produce $x_{01}y_1$, which is partially absorbed on the second pass by T_1 and finally absorbed by

T_2.

Here is an algorithm for 1-rule coverage for RCP. Let *P* be the set of incomplete prime implicants in a running sum. (In a complete algorithm, *P* would consist only of prime implicants, but in an incomplete algorithm the terms in *P* may not all be prime implicants, and we refer to these terms as *incomplete prime implicants*.) The first step is to convert the *n* rules (e.g., $(x < 5)$ $(y < 100)$) into *n* multi-variable terms (e.g., $x_{01}y_{012}$), and we label the terms T_i, for $1 \le i \le n$. We put the first term (corresponding to the first rule) in *P*. The following steps are repeated for the subsequent terms, where the next term is labeled T_i. If T_i is absorbed by a term in *P*, then the rule corresponding to T_i is covered. Otherwise, add T_i to *P*. Since at each step, *P* contains all previous terms, this algorithm performs 1-rule coverage.

The algorithm for using partial absorption for 2-rule coverage is the same as the algorithm for 1-rule coverage except for two differences. First, the partial absorption rule is used instead of the absorption rule. This ensures 2-rule coverage for all coverages that do not reduce a variable to *true*. Second, only the results of applying a multi-variable rule that reduce a variable to *true* are added to *P* (versus all results of applying a multi-variable rule). This ensures 2-rule coverage for all coverages that reduce a variable to *true*.

The algorithm for using partial absorption for 3-rule coverage is the same as the algorithm for 2-rule coverage except for two differences. First, the partial absorption rule is applied for two passes on the terms in *P* (if the term is not partially absorbed on the first pass, then the second pass is unnecessary). This ensures 3-rule coverage for all coverages that do not reduce a variable to *true*. Second, the results of applying a multi-variable rule that reduce a variable to *true*, in addition to being added to *P*, are used with the multi-variable rules and the other terms in *P* (that correspond to the original rules) and those results that reduce a variable to *true* are also added to *P*. This ensures 3-rule coverage when one or two variables are reduced to *true*.

The running time of the 3-rule coverage algorithm is $O(n^4m^3)$ for *n* terms of *m* variables. Note that the size of *P* is *n* plus the number of results having one or two variables reduced to *true*. The number of results having one variable reduced to *true* is $O(n^2m)$, and if all of these combine with the *n* original terms to reduce two variables to *true*, then *P* grows to $O(n^3m^2)$. If, however, the number of terms added to *P* is $O(n)$, which is likely for our applications, then the running time is $O(n^2m)$.

4 Experimental results

In this section we present the results of testing the three methods for solving RCP. We developed two groups of test cases:

- actual classification rules used at GE Capital, and
- random Multi-SAT instances (satisfiability problems where each variable can take on more than two values).

In the following graphs "CPLEX" represents the approach where instances of RCP are reduced to instances of Co-SAT, which are solved using integer programming in the CPLEX package; "OTTER" represents the approach where instances of RCP are reduced to instances of Co-SAT, which are solved using resolution in the OTTER package; and "RULE" represents the approach where instances of RCP are solved with the multi-variable consensus procedure.

The execution times in the graphs include only the time when searching for coverage (i.e., no overhead processing time from reading/interpreting the problem, etc.). We have fewer examples from the OTTER approach because its overhead processing uses days to weeks of CPU time.

4.1 Actual rules used at GE Capital

This group of test cases consists of the actual classification rule sets used at GE Capital. However, we only have 10 test cases.

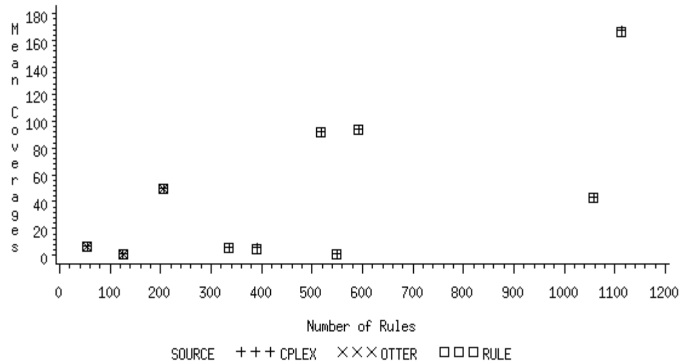

Figure 12: Rule Coverages for CPLEX, OTTER, and Multi-variable Consensus Rule

Figure 12 shows the rule coverages detected by the three approaches, and Figure 13 shows the execution times. The CPLEX and OTTER solutions, being complete, found all of the rule coverages, but they are much slower than the multi-variable consensus procedure. The multi-variable consensus procedure missed only 2 of the 468 covered rules, and it is much faster.

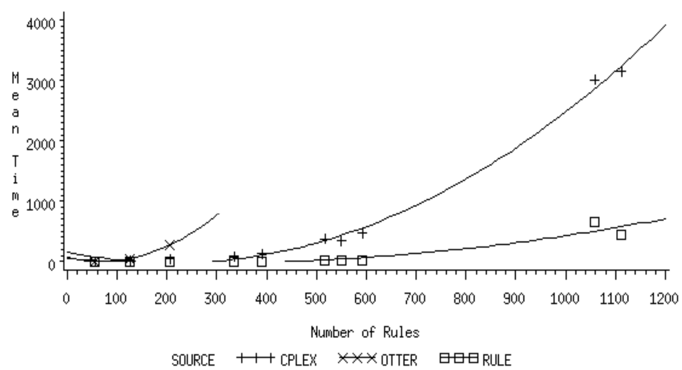

Figure 13: Execution Times for CPLEX, OTTER, and Multi-variable Consensus Rule

4.2 Random Multi-SAT problems

This group of test cases were created to be similar to actual classification rules in order to provide additional insight on the execution times of the three methods. However, no randomly generated instance of Multi-SAT had any covered rules. This indicates that the instances of RCP that occur in practice may not be the harder instances, if the hard instances have a probability of being satisfied equal to 0.5, as in the case of many hard instances of the satisfiability problem.

Figure 14 shows the execution times for the CPLEX and multi-variable consensus procedure solutions. The multi-variable consensus procedure is much faster for these test cases. Due to the extremely long running times of the OTTER solution, the results for this approach are not available yet.

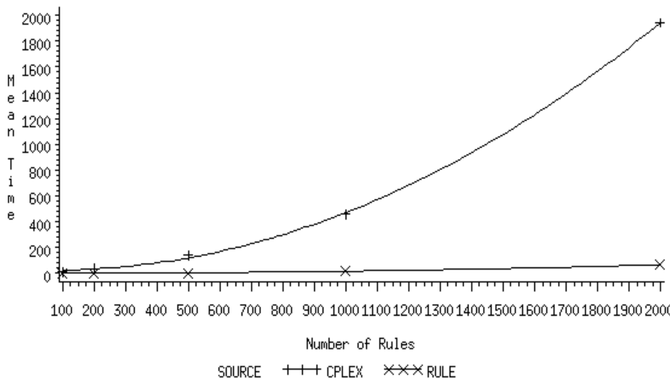

Figure 14: Execution Times for Instances of Random Multi-SAT

5 Bibliography

[Makuch90] W. Makuch. *Optimizing the Collection of Delinquent Consumer Credit*. Ph.D. Thesis, Rensselaer Polytechnic Institute, May, 1990.

[Schneeweiss89] W. Schneeweiss. *Boolean Functions with Engineering Applications and Computer Programs*. Springer-Verlag, New York, N.Y., 1989.

[Hooker88] J. Hooker. A Quantitative Approach to Logical Inference. *Decision Support Systems*, 4:45-69, 1988.

[Garey79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, N.Y., 1979.