The Prime Programming Problem: Formulations and Solution Methods

Montree Jaidee[a], Bismark Singh[a,*]

[a]*School of Mathematical Sciences, University of Southampton, Southampton, SO17 1BJ, UK*

Abstract

We introduce the prime programming problem as a subclass of integer programming. These optimization models impose the restriction of feasible solutions being prime numbers. Then, we demonstrate how several classical problems in number theory can be formulated as prime programs. To solve such problems with a commercial optimization solver, we extend the branch-and-bound procedure of integer programming. Next, in an effort to reduce the computational effort required by this branch-and-bound method, we utilize the special structure of prime numbers to incorporate new branching rules and variable fixing strategies. We present numerical results using a variety of such strategies on a classic conjecture on linear equations in prime numbers. Finally, by employing the concept of the inference dual of an integer program, we show how to perform sensitivity analysis on general prime programs; this allows us to to quickly compute changes in the optimal objective function value for small perturbations in the constraints. We publicly release all of our code to solve a general prime programming model.

Keywords: Integer programming, Prime Numbers, Sensitivity Analysis, Number Theory, Goldbach conjecture

Declarations of interest: none

*Corresponding author

Email address: b.singh@southampton.ac.uk (Bismark Singh)

1. Introduction

Consider the following optimization problem

$$\min \quad f(x, y, z) \tag{1a}$$
$$\text{s.t.} \quad g(x, y, z) \geq 0, \tag{1b}$$
$$x \in \mathbb{P}, y \in \mathbb{R}^+, z \in \mathbb{Z}^+, \tag{1c}$$

where $\mathbb{Z}^+$ denotes the set of non-negative integers, $\mathbb{R}^+$ denotes the set of non-negative real numbers, and $\mathbb{P} = \{2, 3, 5, \ldots\} \subset \mathbb{Z}^+$ denotes the set of prime numbers. Then, model (1) is a standard mixed-integer non-linear program with the inclusion of an additional set of variables, x, restricted to be prime. We call model (1) as a *prime program* (PP). When functions f and g are linear, model (1) is a mixed-integer linear program with prime number constraints. Such models are pervasive in the number theory literature although they are often not formulated in the form of a mathematical program such as optimization model (1). For example, the famous Goldbach's Conjecture rests on proving feasibility of the constraint $2a + 2 = x_1 + x_2$ where $x_1, x_2 \in \mathbb{P}$ for any given $a \in \mathbb{Z}^+$, or its infeasibility for some a. Similarly, the Vinogradov's Theorem says that the constraint $2a + 1 = x_1 + x_2 + x_3$ where $x_1, x_2, x_3 \in \mathbb{P}$ is always feasible for positive integers $a \geq 3$; analogously, the Chen's Theorem says that either of the two constraints $2a = x_1 + x_2$ for some $x_1, x_2 \in \mathbb{P}$ or $2a = x_3 + x_4 \cdot x_5$ for some $x_3, x_4, x_5 \in \mathbb{P}$ is always feasible for a given sufficiently large $a \in \mathbb{Z}^+$. For an introduction to several such problems, see, e.g., Wells (2005).

There is some work at the intersection of mathematical optimization and prime numbers. One stream of work develops heuristics to identify or construct large prime numbers, see, e.g., Knezevic (2021) that uses genetic programming to generate large prime numbers. Dass et al. (2013) study the prime factorization problem using three metaheuristics. Both these works do not formulate an explicit mathematical program, as is standard in the optimization community, to solve the underlying problem using a numerical optimization solver. In contrast, Mehrotra & Pal (2018) develop various objective functions that, when optimized over, solve instances of prime factorization. The work by Kumar (2022) is more general — although without prime number constraints — and, formulates non-linear Diophantine equations as complete mathematical optimization models. For instance, the nonlinear Diophantine $a_1 x_1^p + a_2 x_2^p + \ldots + a_n x_n^p = b$ is reformulated as $\min_x \left(a_1 x_1^p + a_2 x_2^p + \ldots + a_n x_n^p - b\right)^2$ where x is constrained within a feasible region of integers. Our work is in a similar spirit to this: we formulate problems concerning prime numbers as general mathematical optimization models. Such mathematical programs are then easily solvable with standard optimization solvers up to the solver's numerical limits. As such, we introduce the PP in the discrete optimization community.

The structure of this article is as follows. In Section 2, we motivate this work by studying six elementary problems from number theory and formulate them as prime programs. In their current

form, such problems are not solvable with an optimization solver (e.g., CPLEX or Gurobi), since there is no analytical expression to formulate the restriction $x \in \mathbb{P}$. Thus, in Section 3.1, we present a method to solve a PP by modifying the standard branch-and-bound scheme for solving integer problems. Further, we conduct sensitivity analysis for a PP that allows quick solutions of a slightly perturbed problem without resolving it entirely. In Section 4, we present detailed computational experiments and analysis by employing an example of finding a sequence of distinct primes such that the arithmetic mean of any two of these primes is also a prime. We formulate this problem as a PP, and we present additional solution strategies that extend our naive branch-and-bound method. We conclude in Section 5 and provide additional numerical examples and results in the appendix. We provide all of our code to solve a PP at our GitHub repository cited below.

2. Preliminaries

A classic problem in number theory, that is expressible as an integer program, is the computation of the greatest common divisor (gcd) of two integers (Conforti et al., 2014, Chapter 1). For $a, b \in \mathbb{Z}^+$, the gcd of a and b is given by $\gcd(a, b) = \max_z z$ s.t. $\{z$ divides both a and b with $z \in \mathbb{Z}^+\}$. Then, the optimization model

$$\gcd(a, b) = \min_x ax_1 + bx_2 \text{ s.t.}\{ax_1 + bx_2 \geq 1 \text{ with } x_1, x_2 \in \mathbb{Z}^+\} \tag{2}$$

computes the gcd; see, Proposition 1.6 of Conforti et al. (2014) for a proof. Enforcing the additional restriction of solutions, (x_1, x_2), being prime numbers transforms model (2) into a PP. In this section, we provide optimization models for six elementary problems in number theory considered in the classic textbook of Sierpiński (1970). We let $\bar{p}_x$ and $\underline{p}_x$ denote the smallest prime number greater than $x \in \mathbb{R}^+$ and the largest prime number lesser than $x \in \mathbb{R}^+$, respectively.

Problem 1. *Find four triplets of solutions,* (x_1, x_2, x_3)*, of the equation* $x_1^2 + 1 = x_2^2 + x_3^2$ *with* $x_1, x_2, x_3 \in \mathbb{P}$.

Solution. The corresponding optimization model is:

$$\min_x 0 \text{ s.t.}\{x_1^2 + 1 = x_2^2 + x_3^2 \text{ with } x_1, x_2, x_3 \in \mathbb{P}\}. \tag{3}$$

A solution to model (3) provides one such triplet, (x_1^*, x_2^*, x_3^*). In Section 3.1 we describe a procedure to solve this problem, and obtain *an* optimal solution $(x_1^*, x_2^*, x_3^*) = (7, 5, 5)$. To obtain another solution, we add one cutting plane: $x_1 \geq \bar{p}_{x_1^*}$; equivalently, we can add the cutting planes $x_2 \geq \bar{p}_{x_2^*}$, $x_3 \geq \bar{p}_{x_3^*}$, or the weaker cutting plane $x_1 \geq x_1^* + 1$. We continue this process until we obtain four triplets of solutions. Again, using the procedure in Section 3.1, we obtain $(7, 5, 5), (13, 7, 11), (17, 11, 13)$, and $(23, 13, 19)$ as four triplets.

An equivalent optimization formulation is obtained by (i) using the identity $(5z+13)^2+1 = (3z+7)^2+(4z+11)^2, \forall z \in \mathbb{Z}^+$, and (ii) the fact that there are infinitely many primes of the form $az+b$ when $\gcd(a,b)=1$. Then, the following optimization model is equivalent to model (3):

$$\min_{x,z} 0 \text{ s.t.}\{x_1 = 5z+13, x_2 = 3z+7, x_3 = 4z+11 \text{ with } x_1,x_2,x_3 \in \mathbb{P}, z \in \mathbb{Z}^+\}. \tag{4}$$

A variation of this problem is to enforce the largest of these primes, x_1, to be the smallest prime number feasible to model (3). Then, we simply change the objective function of model (3); the corresponding model is

$$\min_{x} x_1 \text{ s.t.}\{x_1^2+1 = x_2^2+x_3^2 \text{ with } x_1,x_2,x_3 \in \mathbb{P}\}. \tag{5}$$

□

Problem 2. *Find all solutions of the equation* $x_1(x_1+1)+x_2(x_2+1) = x_3(x_3+1)$ *with* $x_1,x_2,x_3 \in \mathbb{P}$.

Solution. The corresponding optimization model is:

$$\min_{x} 0 \text{ s.t.}\{x_1(x_1+1)+x_2(x_2+1) = x_3(x_3+1) \text{ with } x_1,x_2,x_3 \in \mathbb{P}\}. \tag{6}$$

Model (6) is a non-linear integer optimization model due to the product of integer variables but is easily linearized with a McCormick envelope, see, e.g., Wolsey (1998). Again, using the procedure in Section 3.1, one such optimal triplet is $(x_1^*, x_2^*, x_3^*) = (2,2,3)$; and, for the next solution, we add a similar cutting plane as in Problem 1. However, in this case the solution is unique as we show below; thus, the computational method of adding cutting planes yields the same solution.

To verify uniqueness of the solution $(2,2,3)$, consider the constraint of model (6) which is equivalent to $x_1(x_1+1) = (x_3-x_2)(x_3+x_2+1)$. From this equation, we immediately have that the prime number x_1 divides at least one of x_3-x_2 or x_3+x_2+1. We distinguish two exclusive cases below.

(i) Consider x_1 divides x_3-x_2. Then, we have $x_1 \leq x_3-x_2$, and it follows that $x_1(x_1+1) \leq (x_3-x_2)(x_3-x_2+1)$. Hence, $x_3+x_2+1 \leq x_3-x_2+1$, which is false since $x_2 \in \mathbb{Z}^+$.

(ii) Consider x_1 divides x_3+x_2+1. Then, there exists a $k \in \mathbb{Z}^+$ such that $x_3+x_2+1 = kx_1$, hence $x_1+1 = k(x_3-x_2)$. If $k=1$, then $x_3+x_2+1 = x_1$ and $x_1+1 = x_3-x_2$. However, this yields $x_1-x_2 = x_3+1$ and $x_1+x_2 = x_3-1$, which cannot be true since $x_2 \in \mathbb{Z}^+$. Now, consider $k \geq 2$ and the identity $2x_2 = (x_3+x_2)-(x_3-x_2)$. We have

$$\begin{aligned} 2x_2 =& (x_3+x_2)-(x_3-x_2) \\ =& (kx_1-1)-(x_3-x_2) \\ =& k[k(x_3-x_2)-1]-1-(x_3-x_2) \end{aligned}$$

$$=(k+1)[(k-1)(x_3-x_2)-1].$$

Since x_2 is prime, the divisors of $2x_2$ are 1, 2, x_2, and $2x_2$. Further, since $k+1 \geq 3$ and divides $2x_2$, either $k+1 = x_2$ or $k+1 = 2x_2$. Again, we distinguish the two exclusive cases below.

(a) Consider $k+1 = x_2$. Then $(k-1)(x_3-x_2) = 3$, hence $(x_2-2)(x_3-x_2) = 3$. Thus, either $x_2 - 2 = 1$ and $x_3 - x_2 = 3$, or $x_2 - 2 = 3$ and $x_3 - x_2 = 1$. In the first case, $x_2 = 3$ and $x_3 = 6$, which is not prime. In the second case, $x_2 = 5$ and $x_3 = 6$, again resulting in a non-prime solution for x_3.

(b) Consider $k+1 = 2x_2$. Then $(k-1)(x_3-x_2) = 2$, hence $2(x_2-1)(x_3-x_2) = 2$. Thus, $x_2 - 1 = x_3 - x_2 = 1$, hence $x_2 = 2$ and $x_3 = 3$. In this case, we obtain the solution $(x_1^*, x_2^*, x_3^*) = (2, 2, 3) \in \mathbb{P}$ which is that computed by our procedure above.

Similar to Problem 1, a variation of this problem is to enforce the largest of these primes, x_3, to be the smallest prime number feasible to model (6). Then, we simply change the objective function of model (6); the corresponding model is

$$\min_x x_3 \text{ s.t.}\{x_1(x_1+1) + x_2(x_2+1) = x_3(x_3+1) \text{ with } x_1, x_2, x_3 \in \mathbb{P}\}. \tag{7}$$

□

Problem 3. *Find all solutions $z \in \mathbb{Z}^+$ with $z+1$, $z+3, z+7, z+9, z+13, z+15 \in \mathbb{P}$.*

Solution. The corresponding optimization model is:

$$\begin{aligned}\min_x 0 \text{ s.t.}\{&x_1 = z+1, x_2 = z+3, x_3 = z+7, x_4 = z+9,\\ &x_5 = z+13, x_6 = z+15 \text{ with } x_1, \ldots, x_6 \in \mathbb{P}, z \in \mathbb{Z}^+\}.\end{aligned} \tag{8}$$

In model (8), the integrality restriction, $z \in \mathbb{Z}^+$, is replaceable with its continuous relaxation, $z \in \mathbb{R}^+$. Again, using the procedure in Section 3.1, one feasible tuple of solutions is $(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*, x_6^*, z^*) = (5, 7, 11, 13, 17, 19, 4)$; for the next solution, we add a similar cutting plane as in Problem 1 and continue this process.

However, again similar to Problem 2, the solution of model (8) is unique. To verify this claim, we note that a feasible solution is of the form $\{x_1^*, x_2^*, x_3^*, x_4^*, x_6^*\} = \{a+1, a+3, a+7, a+9, a+15\}$, where $a \in \mathbb{Z}^+$. This set has exactly one element divisible by 5. Hence, if the smallest number of this set, $a+1$, exceeds 5, then all the numbers cannot simultaneously be prime. Since $z = 1, 2, 3$ are all infeasible, the obtained solution $z^* = 4$ is unique. Indeed, adding a constraint $z \geq 5$ (or, equivalently $x_1 \geq 6$) to the optimization solver renders the model infeasible.

Similar to Problem 1 and Problem 2, a variation of the problem is the following model:

$$\begin{aligned}\min_{x,z} z \text{ s.t.}\{&x_1 = z+1, x_2 = z+3, x_3 = z+7, x_4 = z+9,\\ &x_5 = z+13, x_6 = z+15 \text{ with } x_1,\ldots,x_6 \in \mathbb{P}, z \in \mathbb{R}^+\}.\end{aligned} \tag{9}$$

□

Problem 4. *Find all primes which can be represented both as sums and differences of two primes.*

Solution. The corresponding optimization model is:

$$\min_x 0 \text{ s.t.}\{x_1 = x_2 + x_3, x_1 = x_4 - x_5 \text{ with } x_1,\ldots,x_5 \in \mathbb{P}\}. \tag{10}$$

Here, the decision variables $(x_2^*, x_3^*, x_4^*, x_5^*)$ are auxiliary variables while x_1^* provides the required solution. Then, we obtain $x_1^* = 5$ with our procedure of Section 3.1. For the next solution, we proceed by adding cutting planes similar to Problems 1-3, e.g., $x_1 \geq \bar{p}_{x_1^*}$. However, model (10) is simplified further by a few observations. All prime numbers except 2 are odd, and hence $x_2 + x_3$ is an even number (i.e., not prime) unless exactly one of x_2 and x_3 is 2. Without loss of generality, set $x_3 = 2$. Similarly, we have $x_5 = 2$. We thus have the equivalent optimization model:

$$\min_x x_1 \text{ s.t.}\{x_1 = x_2 + 2, x_1 = x_4 - 2 \text{ with } x_1, x_2, x_4 \in \mathbb{P}\}. \tag{11}$$

Again, the solution to this problem is unique. To verify this claim, let $x_1^* = a$ for some $a \in \mathbb{Z}^+$. Consider the set $\{x_1^*, x_2^*, x_4^*\} = \{a, a-2, a+2\}$. This set has exactly one element divisible by 3 since any integer a is of the form of either $a = 3k$, $a = 3k+1$, or $a = 3k+2$ for some $k \in \mathbb{Z}^+$; in each of the three cases, exactly one element is divisible by 3. Hence, all elements of the set are at most three for them to be prime. Since $x_1 = 1, 2, 3, 4$ are all infeasible, the obtained solution $x_1^* = 5$ is unique. Again, adding a constraint $x_1 \geq 6$ to the optimization solver renders the model infeasible.

□

Problem 5. *Find the five least positive integers, z, such that each of the integers z, $z+1$, $z+2$ is a product of two different primes.*

Solution. The corresponding optimization model is:

$$\min_x z \text{ s.t.}\{z = x_1x_2, z+1 = x_3x_4, z+2 = x_5x_6 \text{ with } x_1,\ldots,x_6 \in \mathbb{P}, z \in \mathbb{Z}^+\}. \tag{12}$$

To ensure that the primes are different, we enforce $x_1 \neq x_2$, $x_3 \neq x_4$, and $x_5 \neq x_6$. Without loss of generality, we add an additional set of constraints: $x_{i+1} \geq x_i + 1, i = 1, 3, 5$. Then, the solution to model (12) provides one feasible value for z; we obtain $(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*, x_6^*, z^*) =$

$(3, 11, 2, 17, 5, 7, 33)$ as the solution using the procedure in Section 3.1. For the next solution, we add a cutting plane: $z \geq z^* + 1$. We repeat this process until we get five solutions for z obtaining $z^* = 33, z^* = 85, z^* = 93, z^* = 141$, and $z^* = 201$, respectively. We are unaware of how many such solutions exist.

□

Problem 6. *Find all integers, z, such that each of the six integers z, $z+2$, $z+6$, $z+8$, $z+12$, $z+14$ are primes.*

Solution. The corresponding optimization model is:

$$\begin{aligned} \min_x 0 \text{ s.t.} \{ & x_2 = x_1 + 2, x_3 = x_1 + 6, x_4 = x_1 + 8, \\ & x_5 = x_1 + 12, x_6 = x_1 + 14 \text{ with } x_1, \ldots, x_6 \in \mathbb{P} \}. \end{aligned} \tag{13}$$

Then, we obtain one feasible solution to model (13) using the procedure in Section 3.1 as $(x_1^*, \ldots, x_6^*) = (5, 7, 11, 13, 17, 19)$. The uniqueness of this solution is verifiable in a method similar to that for Problem 4. Let $x_1^* = a$ for some $a \in \mathbb{Z}^+$. Consider the set $\{x_1^*, x_2^*, x_3^*, x_4^*, x_6^*\} = \{a, a+2, a+6, a+8, a+14\}$ which has exactly one element divisible by 5. Thus, $x_1 \leq 5$ is a valid inequality. Since, $x_1 = 2, 3$ are not feasible (and, $x_1 = \{1, 4\}$ is not prime), the obtained solution $x_1^* = 5$ is unique. □

In Appendix A, we provide three further illustrative examples. Although Problems 1-6 are "toy" problems, they demonstrate how problems in number theory are expressible as PPs. Further, they show how such formulations allow an easy application of a commercial solver, if available; we describe the details of such a solution method in Section 3.1. However, the above formulations mask three subtleties from the original problems of number theory.

(i) First, our optimization problems implicitly upper bound the decision variables, e.g., the solver Gurobi treats numbers larger than 10^{20} as infinite (Gurobi Optimization, LLC, 2024). Thus, the optimization problems are unable to compute solutions larger than this value. Although such large limits are expected to be reasonable for most practical problems, problems in number theory do not have such a restriction. Scaling the numbers is one possible direction to overcome this.

(ii) Second, computing *all* solutions requires running the procedure possibly an exponential number of times; even then we are only guaranteed to obtain solutions up to the solver-imposed implicit upper bound. This exhaustive enumeration is expected as most problems in number theory (and, integer programming) are $\mathcal{NP}$-complete. Even verifying uniqueness of an integer solution of a linear system of equations is $\mathcal{NP}$-complete (Mangasarian & Ferris, 2010). However, when a finite number of solutions are required—such as, Problem 1 and Problem 5—we

expect the solutions to be obtained in a few repetitions, particularly if good cuts are identified. Indeed, in the procedure we describe in Section 3.1, we obtain solutions for these two problems in four and five repetitions, respectively. Further, several problems in number theory typically have solutions that are small prime numbers. For example, in Problem 6 we observe that $z \leq 5$. This is because for $z > 5$ at least one element in the set $\{z+1, z+3, z+7, z+9, z+15\}$ is divisible by 5 and, hence, is not prime.

(iii) Third, the above illustrated optimization procedure rests on a scheme to determine $\bar{p}_x$ given x; this provides an added cutting plane. Unfortunately, there is no analytical formula to determine the prime number immediately proceeding a given integer. However, several theorems from the number theory literature assist in this computation. For example, Bertrand's postulate sets the range of $\bar{p}_x \in (x, 2x-2), \forall x \in \mathbb{P} \setminus \{2, 3\}$, see, e.g., Sylvester (1881). In computational practice, as we also mention above, instead of the cut $x \geq \bar{p}_{x^*}$ we can simply add the relaxed cut $x \geq x^* + 1$ which allows the solver to compute a different optimal solution than x. We build up on this to construct a modified branch-and-bound procedure in the next section.

3. Solving Prime Programs

3.1. Branch-and-bound for Prime Programs

We now consider the PP presented in model (1) with linear functions as follows:

$$z^* = \min \quad cx \tag{14a}$$

$$\text{s.t.} \quad Ax \geq b, \tag{14b}$$

$$x_i \in [2, M_i] \cap \mathbb{P}, \quad i = 1, \ldots, n, \tag{14c}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $M_i \in \mathbb{Z}^+, \forall i = 1, \ldots, n$. Here, M denotes a vector of upper bounds on feasible prime numbers. If such a vector of large enough prime numbers is available, we use that; however, if this is difficult to estimate, we employ large enough integers. In the absence of the prime constraint given by equation (14c), model (14) is solvable via the standard branch-and-bound procedure of integer programming, see, e.g., Wolsey (1998). We now adapt this procedure to solve model (14).

To this end, we first solve the linear programming (LP) relaxation of model (14). If the solution, $x^* = [x_1^*, \ldots x_n^*]$, is prime we stop at the root node itself since the solution is optimal. If at least one element is not prime, we select the element which is farthest from its closest prime to branch on; this is similar to the so-called most fractional rule in integer programming. Given the non-prime branching variable, i, we create two nodes, right and left, that include the additional constraints $x_i \geq \bar{p}_{x_i^*}$ and $x_i \leq \underline{p}_{x_i^*}$, respectively. We store these in a heap, $\mathcal{H}$, in this order. We then explore

the last node (i.e., the left child node) in the heap via a depth-first strategy. We prune nodes if one of the following conditions holds: (a) the node is optimal; i.e., the relaxed solution is prime, (b) the node is infeasible, or (c) the objective function value of the relaxed solution at that node is not better than that of the incumbent; i.e., the node is pruned if the objective function value of the node is greater than the incumbent's objective function value in a minimization problem. When a node is pruned, we remove it from the heap. We terminate the algorithm when all nodes are explored or pruned; i.e., $\mathcal{H} = \phi$. We note that solving the linear programming (LP) relaxation with such a branching procedure is sufficient as it partitions the feasible space into disjoint spaces without cutting off any feasible solution. We illustrate this branch-and-bound procedure with the following numerical example.

$$\begin{aligned} \min \quad & -x_1 - 4x_2 && \text{(15a)} \\ \text{s.t.} \quad & -3x_1 - x_2 \geq -2000, && \text{(15b)} \\ & -x_1 - 5x_2 \geq -450, && \text{(15c)} \\ & x_1, x_2 \in [2, 100] \cap \mathbb{P}. && \text{(15d)} \end{aligned}$$

Figure 1 presents the branch-and-bound tree for model (15) that we summarize here; nodes marked with white are pruned below while those in black are further branched. We first strengthen the upper bound of $M = 100$ in constraint (15d) with $M = \underline{p}_{100} = 97$. Then, the LP solution at the root node is $x^* = [97, 70.6]$. Since x_1 is prime, we branch on x_2. We identify $\bar{p}_{70.6} = 71$ (right branch) and $\underline{p}_{70.6} = 67$ (left branch). We proceed to the left child-branch first (marked "2"); it is pruned, identifying an incumbent solution. We then backtrack to the right branch at node "3", which is the next element in the heap, and continue the process. Node "5" is pruned via optimality and we backtrack to node "4" to its right branch. Continuing, we identify a new incumbent solution at node "8". The right branch at node "9" identifies a solution that is not prime which is then pruned by bound. Backtracking we proceed first to the right branch of node "6" (i.e., node "10") and then to the right branch of node "3" (i.e., node "11") — both are pruned by infeasibility. The optimal solution is then $x^* = [83, 73]$ with an optimal objective function value of -375. The enumeration tree contains 11 nodes, six of which are leaf nodes.

3.2. Sensitivity Analysis for Prime Programs

We now describe a procedure to conduct sensitivity analysis on a linear PP. This allows us to compute changes in the optimal objective function value for small perturbations in the problem parameters without re-solving the entire program. To this end, we employ the technique of the inference dual of an integer program as proposed in Dawande & Hooker (2000). Consider the following problem that formulates a perturbation in model (14).

$$z^*_\delta = \min \quad (c + c_\delta)x \qquad \text{(16a)}$$

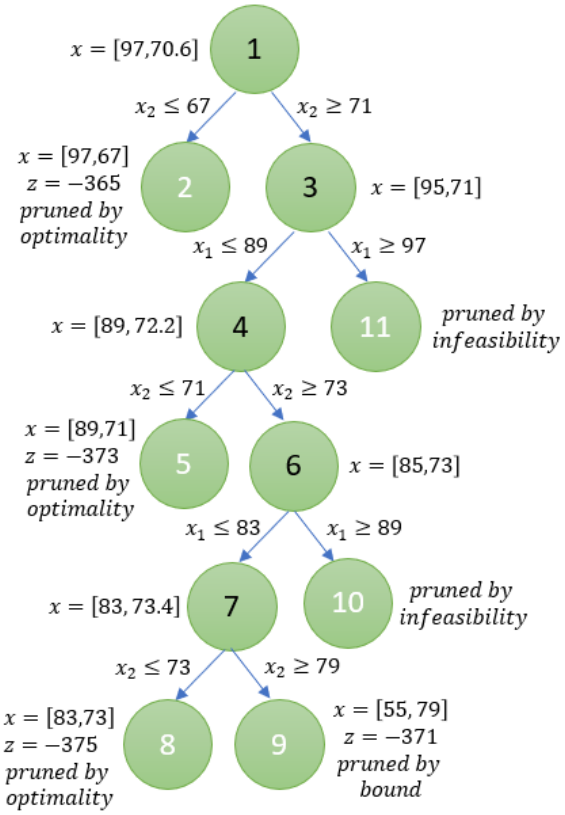


Figure 1: Enumeration tree for the prime branch-and-bound procedure applied to model (15). The six nodes marked in white are leaf nodes that are pruned below for reasons indicated, while the other five nodes are further branched on. For details, see Section 3.1.

$$\text{s.t.} \quad (A + A_\delta)x \geq b + b_\delta, \tag{16b}$$

$$x_i \in [2, M_i] \cap \mathbb{P}, \quad i = 1, \ldots, n. \tag{16c}$$

In the absence of constraint (16c), the work in Dawande & Hooker (2000) provides a sufficient condition ensuring $z^* - \Delta \leq z^*_\delta$, where $\Delta \geq 0$, by examining all the leaf nodes of the branch-and-bound tree for model (14). At a leaf node p of the branch-and-bound tree, we let $\bar{x}^p$ and $\underline{x}^p$ denote upper and lower bounds for x, and λ^p denote the optimal value of the dual variables of constraint (14b). Further, we let z_p^{UB} denote the best upper bound on the optimal objective function value obtained up to node p in the tree; if no such bound is available, we set $z_p^{UB} = \infty$. We distinguish three cases for the three types of leaf nodes.

(i) If the leaf node is pruned by infeasibility, we let $q^p = \lambda^p A$, $q^p_\delta = \lambda^p A_\delta$, and $z_p = \varepsilon$, where ε is a small positive quantity.

(ii) If the leaf node is pruned by bound, we let $q^p = \lambda^p A - c$, $q^p_\delta = \lambda^p A_\delta - c_\delta$, and $z_p = z_p^{UB} - \Delta$.

(iii) If the leaf node is pruned by optimality, we let $q^p = \lambda^p A - c$, $q^p_\delta = \lambda^p A_\delta - c_\delta$, and $z_p = \bar{z}_p - \Delta$ where $\bar{z}_p$ is the objective function value at that node.

Then, the following theorem relates the objective functions of model (14) and its perturbation in model (16).

Theorem 1. *Consider model (14) and its perturbation in model (16). Let q_p, q^p_δ, z_p, and λ^p be as defined above. Then, $z^* - \Delta \leq z^*_\delta$ holds for all $\Delta \geq 0$ if there exist $\bar{q}^p_j, j = 1, \ldots, n$ satisfying*

$$\sum_{j=1}^{n} \left((q^p_j + q^p_{\delta\, j}) \underline{x}^p_j + \bar{q}^p_j (\bar{x}^p_j - \underline{x}^p_j) \right) \leq \lambda^p (b + b_\delta) - z_p, \tag{17a}$$

$$\bar{q}^p_j \geq q^p_j + q^p_{\delta\, j}, \;\; \bar{q}^p_j \geq 0, \quad j = 1, \ldots, n, \tag{17b}$$

for each leaf node p of the branch-and-bound tree of model (14).

Proof. See Dawande & Hooker (2000) and Appendix B. □

The proof of Theorem 1 mirrors that in Dawande & Hooker (2000) for mixed-integer programs (i.e., in the absence of prime number constraints); hence, we reserve it for the appendix. Although Theorem 1 and the results in Dawande & Hooker (2000) provide only a sufficient, and not necessary, condition, there is value in such sensitivity analysis as we demonstrate next. In Problem 7, we consider an illustrative example inspired by the Goldbach conjecture.

Problem 7. *Find the smallest prime number, x_1, satisfying the condition $2k + 2 = x_1 + x_2$ for a given $k \in \mathbb{Z}^+$ where $x_2 \in \mathbb{P}$.*

Solution. The following PP formulates Problem 7.

$$z^* = \min_{x_1,x_2} \quad x_1 \tag{18a}$$

$$\text{s.t} \quad x_1 + x_2 \geq 2k + 2, \tag{18b}$$

$$-x_1 - x_2 \geq -2k - 2, \tag{18c}$$

$$x_1, x_2 \in [2, M] \cap \mathbb{P}. \tag{18d}$$

where M is a large enough upper bound. □

Now consider a perturbation in Problem 7: find the smallest prime number, x_1, satisfying the condition $2k + 2 + k_\delta = x_1 + x_2$ for given k where k_δ is an even integer and $x_2 \in \mathbb{P}$. In other words, we change the right hand sides of equations (18b) and (18c). Then, we need to solve the following PP:

$$z^*_\delta = \min_{x_1,x_2} \quad x_1 \tag{19a}$$

$$\text{s.t} \quad x_1 + x_2 \geq 2k + 2 + k_\delta, \tag{19b}$$

$$-x_1 - x_2 \geq -2k - 2 - k_\delta, \tag{19c}$$

$$x_1, x_2 \in [2, M] \cap \mathbb{P}. \tag{19d}$$

Following the notation of model (16), in model (19) we have $A_\delta \leftarrow 0, c_\delta \leftarrow 0$, and $b_\delta = [k_\delta, -k_\delta]$. For a numerical demonstration, consider $k = 1699$. Figure 2 presents the branch-and-bound tree for this instance of model (7); we have, $z^* = 11$. The enumeration tree contains five nodes, three of which are leaf nodes (marked in white).

(i) Consider node 3. We have $\underline{x}^3 = [2, 2], \bar{x}^3 = [7, 3391], z_3 = \varepsilon, \lambda^3 = [0, 1]$. Then, $q^3 = [1, 1], q^3_\delta = [0, 0]$. In order to satisfy the sufficient condition of Theorem 1 at this node, we require a solution $\bar{q}^3_1, \bar{q}^3_2$ such that $4 + 5\bar{q}^3_1 + 3389\bar{q}^3_2 \leq 3400 + k_\delta - \varepsilon$ and $\bar{q}^3_1, \bar{q}^3_2 \geq 1$. By setting $\bar{q}^3_1, \bar{q}^3_2 = [1, 1]$ we obtain $k_\delta \geq -2 + \varepsilon$.

(ii) Consider node 4. We have $\underline{x}^4 = [11, 2], \bar{x}^4 = [M, 3391], z_4 = 11 - \Delta, \lambda^4 = [0, 0]$. Then, $q^4 = [-1, 0], q^4_\delta = [0, 0]$. In order to satisfy the sufficient condition of Theorem 1 at this node, we require a solution $\bar{q}^4_1, \bar{q}^4_2$ such that $-11 + (M - 11)\bar{q}^4_1 + 3389\bar{q}^4_2 \leq -11 + \Delta$ and $\bar{q}^4_1, \bar{q}^4_2 \geq 0$. By setting $\bar{q}^4_1, \bar{q}^4_2 = [0, 0]$ we obtain $\Delta \geq 0$.

(iii) Consider node 5. We have $\underline{x}^5 = [2, 3407], \bar{x}^5 = [M, M], z_5 = \varepsilon, \lambda^5 = [1, 0]$. Then $q^5 = [-1, -1], q^5_\delta = [0, 0]$. In order to satisfy the sufficient condition of Theorem 1 at this node, we require a solution $\bar{q}^5_1, \bar{q}^5_2$ such that $-3409 + (M - 2)\bar{q}^5_1 + (M - 3407)\bar{q}^5_2 \leq -3400 - k_\delta - \varepsilon$ and $\bar{q}^5_1, \bar{q}^5_2 \geq 0$. By setting $\bar{q}^5_1, \bar{q}^5_2 = [0, 0]$ we obtain $k_\delta \leq 9 - \varepsilon$.

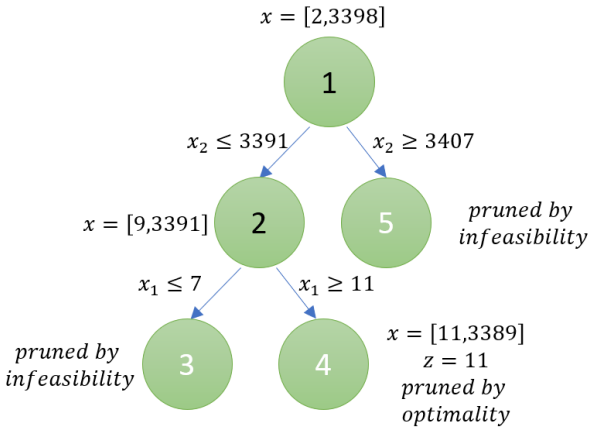


Figure 2: Enumeration tree for the prime branch-and-bound procedure applied to model (18) with $k = 1699$. The three nodes marked in white are leaf nodes that are pruned below for reasons indicated, while the other two nodes are further branched on. For details, see Section 3.2.

Thus, we conclude that $z^*_\delta \geq 11 - \Delta$ holds for all $\Delta \geq 0$ if $k_\delta \in (-2, 9)$. The optimal solution of model (18) is $x^* = [11, 3389]$. We now illustrate the value of sensitivity analysis for PP via the following arguments.

Since k_δ is even, the set of small-enough allowed values is $\{2, 4, 6, 8\}$. Now, consider, $k_\delta = 2$ and the corresponding optimization model: $\min_x x_1$ s.t.$\{x_1 + x_2 = 3402$ with $x_1, x_2 \in \mathbb{P}\}$. Instead of re-solving this model, we simply check the feasibility of the optimal solution of model (18) for model (19). From the condition, $x^*_1 \geq 11 - \Delta, \forall \Delta \geq 0$ it is sufficient to check feasible values of x_1 that are at least 11. Then, since $[x_1, x_2] = [11, 3391]$ is feasible for model (19), it is also optimal; hence, the optimal objective function value is 11. We thus obtain the optimal solution without any additional computation (except checking that 3391 is a prime number). In contrast, the branch-and-bound tree for this instance of model (19) has three nodes. Next, consider $k_\delta = 4$ and the corresponding optimization model: $\min_x x_1$ s.t.$\{x_1 + x_2 = 3404$ with $x_1, x_2 \in \mathbb{P}\}$. Then, $[x_1, x_2] = [11, 3393]$ is infeasible. The next solution $[x_1, x_2] = [13, 3391]$ is feasible; hence, the optimal objective function value is 13. We obtain this solution in two evaluations. In contrast, the branch-and-bound tree for this instance of model (19) again has three nodes. Similarly, for $k_\delta = 6$ and $k_\delta = 8$, we obtain the optimal objective function value of 17 in only three additional evaluations. Thus, following Theorem 1, we save computational effort by not considering prime numbers below 11; i.e., we obviate checking the primes 3, 5, and 7. In Appendix C, we provide another problem which is also inspired by the Goldbach conjecture to illustrate the value of Theorem 1.

4. Solution Strategies for Prime Programs

In Section 3.1, we present a branch-and-bound strategy to solve prime programs. This branch-and-bound procedure is generic and capable of solving any PP; however, the computational effort could be prohibitive. Similar to integer programming, certain prime programs allow us to exploit their structural properties to obtain solutions in a computationally efficient way. Problems 1-6 present examples of how insights from number theory can be used to facilitate solutions. In this section, we exploit such properties of the presented branch-and-bound procedure by revisiting a classic problem in the number theory literature and employing it as a numerical case-study.

We carry out all computational experiments on a computing cluster with an Intel Xeon Gold 6138 2.0GHz processor with 192 GB of RAM and Gurobi version 10.0.3 using Python 3.8.3. We require only a LP solver, and use Gurobi's default LP optimality tolerance of 10^{-6}. We provide the corresponding branch-and-bound code at our GitHub page. As we demonstrate next, solving the problem we consider —given an upper bound of the considered feasible solution—via the branch-and-bound method of Section 3.1 is computationally challenging.

4.1. Linear Equations in Primes

Conjecture 1 (Pomerance et al. (1988))**.** *There exist n distinct prime numbers $x_1 < x_2 < \ldots < x_n$ such that the average of any two of these primes, given by $\frac{1}{2}(x_i + x_j), \forall i, j = 1, \ldots, n, i \neq j$, is also a prime number for $n \geq 2$.* □

Conjecture 1 is proposed in Pomerance et al. (1988) and is proven in Granville (1990) assuming the so-called k-Tuple Conjecture of Hardy and Littlewood (Hardy & Littlewood, 1923) is true. A solution of Conjecture 1 is not unique for a given n, e.g., consider $n = 3$ then $[3, 7, 19]$ and $[5, 17, 41]$ are both valid solutions. We are only aware of solutions of Conjecture 1 until $n = 12$; for $n = 12$ a solution is provided in Balog (1992): $[5, 17, 521, 42281, 138461, 195137, 204137, 221537, 363497, 367001, 414737, 434717]$.

To solve Conjecture 1, we formulate a sequence of PPs of increasing fidelity. These PPs take as input both n and an upper bound, M, for the considered set of prime numbers. We begin with a naive PP as follows.

$$\min_{x,y} \quad 0 \tag{20a}$$

$$\text{s.t.} \quad 2y_{ij} = x_i + x_j, \qquad 1 \leq i < j \leq n, \tag{20b}$$

$$x_{i+1} \geq x_i + 1, \qquad 1 \leq i \leq n-1, \tag{20c}$$

$$x_i \in [2, M] \cap \mathbb{P}, \qquad 1 \leq i \leq n, \tag{20d}$$

$$y_{ij} \in [2, M] \cap \mathbb{P}, \qquad 1 \leq i < j \leq n. \tag{20e}$$

An optimal solution of model (20), if it exists, provides a vector $[x]_n$, with each element below M, that satisfies Conjecture 1. Here, constraints (20b) and (20e) ensure that the average of any two

components of $[x]$ is also prime, where y_{ij} is the average of x_i and x_j for $1 \leq i < j \leq n$. Constraint (20c) ensures that the solutions are distinct primes, while constraint (20d) restricts x to be prime. Then, from Section 3.1, model (20) is a PP.

We now tighten the formulation in model (20) with a few observations. First, constraint (20b) requires the elements of x to be odd primes; thus, we exclude the prime 2 from the optimal solution of model (20). We make two further observations: (i) prime numbers differ by at least 2, and (ii) we require at least one prime number between x_i and x_j. Thus, we tighten constraint (20c) and have a more efficient model given by the PP below.

$$\min_{x,y} \quad 0 \tag{21a}$$

$$\text{s.t.} \quad 2y_{ij} = x_i + x_j, \qquad 1 \leq i < j \leq n, \tag{21b}$$

$$x_{i+1} \geq x_i + 4, \qquad 1 \leq i \leq n-1, \tag{21c}$$

$$y_{ij} \geq x_i + 2, \qquad 1 \leq i < j \leq n, \tag{21d}$$

$$x_j \geq y_{ij} + 2, \qquad 1 \leq i < j \leq n, \tag{21e}$$

$$x_i \in [3, M] \cap \mathbb{P}, \qquad 1 \leq i \leq n, \tag{21f}$$

$$y_{ij} \in [5, M] \cap \mathbb{P}, \qquad 1 \leq i < j \leq n. \tag{21g}$$

n	M	Nodes	Time	Solution
2	1000	7	-	[3, 7]
3	1000	31	-	[3, 7, 919]
4	1000	119	-	[3, 7, 19, 139]
5	1000	269	-	[3, 7, 19, 139, 859]
6	1000	189049	90	[3, 11, 23, 71, 191, 443]
7	1000	159473	82	[5, 17, 41, 101, 257, 461, 521]
8	1000	1286503	821	[5, 17, 41, 101, 257, 521, 761, 881]
9	1000	4080930	3000	ϕ
	1000	8209983	6000	ϕ
	10000	3658991	3000	ϕ
	10000	7353380	6000	ϕ

Table 1: Computational results on solving model (21) using the `Naive` branch-and-bound solution strategy presented in Section 3.1. Here, n and M are inputs to model (21), Nodes denotes the number of explored nodes in the branch-and-bound tree, and Time denotes the computational time in seconds. Entries of "-" in the time column denote the solution is obtained practically instantaneously. The Solution column presents an optimal solution found with the given parameters, here the entry ϕ in the rows with $n = 9$ denotes that no solution is obtained; increasing the value of M and Time still does not provide a solution.

Table 1 presents our computational results for model (21) with $M = 1000$. We solve model (21) with the branch-and-bound method presented in Section 3.1. Solutions for $n \leq 5$ are obtained practically instantly, while for $n = 6, 7, 8$ the solution times are slightly longer. However, no solution is obtained for $n = 9$ within 50 minutes and neither is the model proven infeasible. Increasing the time limit (to allow exploration of a greater number of nodes of the branch-and-bound tree) to 100 minutes or increasing M (to allow exploration of a greater number of potentially feasible prime numbers) to 10000 still provides no feasible solution. Each branch-and-bound node solves a LP, and this computation is highly effective benefiting from modern LP solvers; for $n = 6, 7, 8, 9$, the computational time per 1000 nodes is only marginally over half a second. Thus, we now explore tailored strategies to determine the branching variables as well as fixing certain variables to solve model (21). We refer to the branching strategy of Section 3.1 as the `Naive` strategy, and begin with variable fixing strategies.

4.2. Variable-fixing Strategies

Computing a solution for model (21) becomes increasingly computationally difficult as n increases; however, the solution $[x]_n$ includes several components from the solution $[x]_{n-1}$. This observation motivates a variable fixing strategy where the first $n-1$ components of $[x]_n$ are fixed from the previously obtained solution. We then solve model (21) for the largest missing element alone. We denote such a strategy of fixing the decision variables as `SelectAll`.

Routine 1 `fixing strategy` (name, arg)

Input: *name* = {`Naive`, `SelectAll`, `ExcludeOne`, `ExcludeTwo`} of chosen fixing strategy; a vector $[\bar{x}]$ of dimension $n-1$, additional arguments arg of appropriate dimension. Below, $a \leftarrow \phi$ denotes no value is assigned to a.

Output: a vector $[x]$ of dimension n.

1: **if** name = `Naive`

2: $\quad x_i \leftarrow \phi$, $\forall i = 1, \ldots, n$.

3: **if** name = `SelectAll`

4: $\quad x_i \leftarrow \bar{x}_i, i = 1, \ldots, n-1$; $x_n \leftarrow \phi$.

5: **if** name = `ExcludeOne`

6: $\quad (l) \leftarrow$ arg.

7: $\quad x_i \leftarrow \bar{x}_i, i = 1, \ldots, l-1$; $x_i \leftarrow \bar{x}_{i+1}, i = l, \ldots, n-2$; $x_i \leftarrow \phi, i = n-1, n$.

8: **if** name = `ExcludeTwo`

9: $\quad (l_1, l_2) \leftarrow$ arg, with $l_1 < l_2$.

10: $\quad x_i \leftarrow \bar{x}_i, i = 1, \ldots, l_1 - 1$; $x_i \leftarrow \bar{x}_{i+1}, i = l_1, \ldots, l_2 - 2$; $x_i \leftarrow \bar{x}_{i+2}, i = l_2 - 1, \ldots, n-3$; $x_i \leftarrow \phi, i = n-2, \ldots, n$.

The `SelectAll` strategy might result in infeasibility of model (21) due to too many variables

being fixed. Thus, rather than fixing all the first $n-1$ components of x, we selectively exclude some elements from fixing, e.g., those near their upper bound. To this end, we define two other strategies — `ExcludeOne` and `ExcludeTwo` — that exclude one and two elements from fixing while keeping the two and three largest elements unassigned, respectively. Routine 1 summarizes the three variable fixing strategies, while Example 1 illustrates them numerically; here, the quantity arg denotes the input arguments to the respective strategies. In Section 4.4, we provide the benefit of these three enhanced strategies over the `Naive` strategy.

Example 1. *An illustration of the four variable fixing strategies of Routine 1.*

Let $n = 7$ and consider a known solution, $\bar{x}$, for $n = 6$ with $\bar{x} = [7, 19, 67, 127, 547, 607]$. From the seven positions of x to determine, we fix at most six positions from the known solution $\bar{x}$. In the `Naive` strategy we do not fix any variables. Our remaining three strategies are as follows.

(i) In the `SelectAll` strategy, all six elements are fixed. Then, we have $x = [7, 19, 67, 127, 547, 607, \phi]$.

(ii) In the `ExcludeOne` strategy, we have one element, l, excluded as determined by its input argument while the other five elements are fixed. Consider $l = 3$. Then, we have $x = [7, 19, 127, 547, 607, \phi, \phi]$.

(iii) In the `ExcludeTwo` strategy, we have two elements, l_1 and l_2, that are excluded as determined by its input argument, while the other four elements are fixed. Consider $(l_1, l_2) = (3, 4)$. Then, we have $x = [7, 19, 547, 607, \phi, \phi, \phi]$.

□

4.3. Branching Strategies

By examining the solutions in Table 1, we observe that the solutions for $n = 2$ to 6 are similar; this observation motivated the variable fixing strategies of Section 4.2. However, none of the elements of the solutions from $n = 2$ to 6 are present in the solutions for $n = 7$ and $n = 8$. In this section, we present a different strategy motivated by integer number theory. Unlike integer programming, prime numbers are not spaced equally. We thus investigate strategies related to the remainder of an integer when divided by another integer. In this section, we consider prime numbers that are at least 5; checking whether 3 forms part of the solution is easily done as a post-processing step to the optimization. We begin with two simple lemmas.

Lemma 1. *Any prime number $x \geq 5$ is expressible as $x = 12z + k$ for some $z \in \mathbb{Z}$ and some $k \in \{1, 5, 7, 11\}$.*

Proof. Any integer x is expressible as $x = 12z + k$ for some $z \in \mathbb{Z}$, where $0 \leq k \leq 11$ is the remainder of x when divided by 12. When $k = 0$, $x = 12z$, which is divisible by 12; thus, x is not prime. Similarly, if $k = 2, 3, 4, 6, 8, 9, 10$, then x is divisible by $2, 3, 4, 6, 4, 3, 2$, respectively; hence,

Routine 2 `branching strategy` (name, LP solution at node, arg)

Input: *name* ={`Naive`, `Modulo`} of chosen branching strategy; arguments arg of the chosen branching strategy, arg $= \phi$ if *name* = `Naive`, arg $= \{1, 5, 7, 11\}$ if *name* = `Modulo`; $(x^*, y^*) \leftarrow$ a solution of the linear programming relaxation of model (21) at a given node.

Output: constraints for right and left child nodes of the input node.

if name = `Naive`
  for $i = 1, \ldots, n$
    if $x_i^* \notin \mathbb{P}$
      $distX_i \leftarrow \min\{\bar{p}_{x_i^*} - x_i^*, x_i^* - \underline{p}_{x_i^*}\}$.
    else
      $distX_i \leftarrow 0$.
    for $j = i + 1, \ldots, n$
      if $y_{ij}^* \notin \mathbb{P}$
        $distY_{ij} \leftarrow \min\{\bar{p}_{y_{ij}^*} - y_{ij}^*, y_{ij}^* - \underline{p}_{y_{ij}^*}\}$.
      else
        $distY_{ij} \leftarrow 0$.
  $indX \leftarrow \arg\max_i distX_i$; $indY \leftarrow \arg\max_{ij} distY_{ij}$.
  if $distX_{indX} > distY_{indY}$
    return right and left nodes that contain additional constraints $x_{indX} \geq \bar{p}_{x_{indX}^*}$ and $x_{indX} \leq \underline{p}_{x_{indX}^*}$, respectively.
  else
    return right and left nodes that contain additional constraints $y_{indY} \geq \bar{p}_{y_{indY}^*}$ and $y_{indY} \leq \underline{p}_{y_{indY}^*}$, respectively.
if name = `Modulo`
  $k \leftarrow$ arg.
  $k_0 \leftarrow k \mod 6$.
  for $i = 1, \ldots, n$
    if $(x_i^* \notin \mathbb{P})$ or $(x_i^* \mod 12 \neq k)$
      $distX_i \leftarrow \min\{\bar{p}_{x_i^*}(12, k) - x_i^*, x_i^* - \underline{p}_{x_i^*}(12, k)\}$.
    else
      $distX_i \leftarrow 0$.
    for $j = i + 1, \ldots, n$
      if $(y_{ij}^* \notin \mathbb{P})$ or $(y_{ij}^* \mod 6 \neq k_0)$
        $distY_{ij} \leftarrow \min\{\bar{p}_{y_{ij}^*}(6, k_0) - y_{ij}^*, y_{ij}^* - \underline{p}_{y_{ij}^*}(6, k_0)\}$.
      else
        $distY_{ij} \leftarrow 0$.
  $indX \leftarrow \arg\max_i distX_i$; $indY \leftarrow \arg\max_{ij} distY_{ij}$.
  if $distX_{indX} > distY_{indY}$
    return right and left nodes that contain additional constraints $x_{indX} \geq \bar{p}_{x_{indX}^*}(12, k)$ and $x_{indX} \leq \underline{p}_{x_{indX}^*}(12, k)$, respectively.
  else
    return right and left nodes that contain additional constraints $y_{indY} \geq \bar{p}_{y_{indY}^*}(6, k_0)$ and $y_{indY} \leq \underline{p}_{y_{indY}^*}(6, k_0)$, respectively.

it is not prime. Therefore, for a given prime $x \geq 5$, it is expressible only in the form $x = 12z + k$ for some $z \in \mathbb{Z}$ and some $k \in \{1, 5, 7, 11\}$. For example, the prime number $23 = 12 \cdot 1 + 11$. □

Lemma 2. *In an optimal solution for model (21), the prime number y_{ij} is given by the average of two prime numbers, x_i and x_j, both of which leave the same remainder when divided by 12. Hence, y is of the form $6z + k$ for some $z \in \mathbb{Z}$, and some $k \in \{1, 5\}$.*

Proof. From Lemma 1, consider $x_1 = 12z_1 + k_1$ and $x_2 = 12z_2 + k_2$ for some $z_1, z_2 \in \mathbb{Z}$ and some $k_1, k_2 \in \{1, 5, 7, 11\}$, and $x_1, x_2 \geq 5$. Table 2 provides the average of x_i and x_j for different values of k_1 and k_2. The average cannot be prime unless $k_1 = k_2$. This completes the first part of the lemma. Then, from constraint (21) (equivalently, from the diagonal entries of Table 2), y_{ij} is of the form $y_{ij} = 6z_{ij} + k$ for some $z_{ij} \in \mathbb{Z}$ where $k \in \{1, 5\}$. □

x_1 \ x_2	$12z_2 + 1$	$12z_2 + 5$	$12z_2 + 7$	$12z_2 + 11$
$12z_1 + 1$	$6(z_1 + z_2) + 1$	$3(2z_1 + 2z_2 + 1)$	$2(3z_1 + 3z_2 + 2)$	$6(z_1 + z_2)$
$12z_1 + 5$	$3(2z_1 + 2z_2 + 1)$	$6(z_1 + z_2) + 5$	$6(z_1 + z_2 + 1)$	$2(3z_1 + 3z_2 + 4)$
$12z_1 + 7$	$2(3z_1 + 3z_2 + 2)$	$6(z_1 + z_2 + 1)$	$6(z_1 + z_2 + 1) + 1$	$3(2z_1 + 2z_2 + 3)$
$12z_1 + 11$	$6(z_1 + z_2)$	$2(3z_1 + 3z_2 + 4)$	$3(2z_1 + 2z_2 + 3)$	$6(z_1 + z_2 + 1) + 5$

Table 2: Averages of pairs of prime numbers of the form $x_1 = 12z_1 + k$ and $x_2 = 12z_2 + k$ for $k = \{1, 5, 7, 11\}$. For details, see Section 4.3 and Lemma 2.

Lemmas 1 and 2 allow an improvement in the branching behavior of model (21). A straightforward way of doing so is to add additional integer variables to model (21). However, instead we change the way the branch-and-bound method constructs the disjunctive branches. Consider $x = 24.7$ at some node of the branch-and-bound tree of the `Naive` strategy. Then, the two branches are $x \leq 23$ and $x \geq 29$. Now, let $\bar{p}_x(m, k)$ denote the smallest prime number of the form $mz + k$ for some $z \in \mathbb{Z}$ which is greater than $x \in \mathbb{R}^+$, and $\underline{p}_x(m, k)$ denote the largest prime number of the form $mz + k$ for some $z \in \mathbb{Z}$ which is less than $x \in \mathbb{R}^+$. We add the additional constraint for the two branching nodes as $x_i \geq \bar{p}_{x_i^*}(m, k)$ and $x_i \leq \underline{p}_{x_i^*}(m, k)$, respectively, for $m = 12$ and $k = 1$ (from Lemma 2). Then, the two branches under this new branching rule are $x \leq \underline{p}_{24.7}(12, 1) = 13$ and $x \geq \bar{p}_{24.7}(12, 1) = 37$. With the `Naive` strategy, we need to introduce a new integer variable $z \in \mathbb{Z}$ such that $x = 12z + 1$ and then branch on one variable at a time. Branching on the variable $x = 24.7$ leads to the two branches: $x \leq 23$ and $x \geq 29$. In contrast, branching on the variable $z = 1.975$ leads to the two branches: $z \leq 1$ and $z \geq 2$ which are equivalent to $x \leq 13$ and $x \geq 25$. The advantages of changing the branching method are twofold. First, it provides stronger constraints than those obtained from naive branching method since the solution is always a prime of the form we are considering. Second, it is not necessary to introduce additional variables into the

model. We call such a branching strategy as `Modulo` with the argument arg denoting the value of $k \in \{1, 5, 7, 11\}$. Routine 2 summarizes the `Modulo` branching strategies.

4.4. Computation Results

In this section, we present our computational experiments to solve model (21) using the fixing strategy described in Routine 1 and the branching strategy described in Routine 2. However, rather than directly use them as solution procedures, we begin with a need for an iterative algorithm for solving model (21) that takes an input both these strategies. For example, consider a moderate sized instance of model (21) with $n = 5$ and $M = 3000$ that we solve with the `ExcludeTwo` fixing strategy and the `Modulo` branching strategy. From the total of six instances each for arg = 1, 5, 7, and 11, we obtain feasible solutions in only two, one, two, and three instances, respectively; the remaining instances are all infeasible. However, increasing the upper bounds results in all instances being feasible within two seconds. Similarly, there are easily identifiable instances where neither feasibility nor infeasibility is proved within a given time limit; Example S1 in Appendix D provides a numerical instance that shows the number of such intractable instances is reduced when the time limit is increased. Further, in Example S2 and S3 of Appendix D we demonstrate the computational benefits of the variable fixing and branching strategies over their naive counterparts. Since we know model (21) is feasible until at least $n = 12$ given a sufficiently large time limit and upper bounds, we initialize solution procedures with a small enough upper bound and iteratively increase the bound if an instance is proven infeasible. Algorithm 3 presents the complete scheme we employ to solve model (21). We illustrate the working of the algorithm in Example 2. In the interest of space, we present only the most relevant computational results here while we reserve the complete set of results for Appendix D.

Example 2. *Consider an instance of model* (21) *with inputs* $n = 3$, $M = 1000$, $T = 600$ *seconds,* $t = 300$ *seconds,* $\alpha = 0.2$, *`fixing strategy` =* `SelectAll` *with* arg $= \phi$, *and `branching strategy` =* `Modulo` *with* arg $= 1$ *applied to each node in Algorithm 3.*

We initialize Algorithm 3 with $k \leftarrow 0$, $m \leftarrow 1$, *time* $\leftarrow 0$, and $[x^0]_0 \leftarrow []$. In the first iteration, we apply the `fixing strategy` = `SelectAll`, resulting in $[x^0]_1 \leftarrow [\phi]$. Using this fixed solution, we solve model (21) with the `Modulo` with arg = 1 branching strategy to obtain a feasible solution in Step 4. Then, $k \leftarrow 1$ and *time* is updated; currently, the fixed solution is $[x^1]_1 = [13]$ and $m \leftarrow 2$. In the second iteration, we apply the `fixing strategy` again, now yielding $[x^1]_2 = [13, \phi]$. Solving the model, we obtain a solution of $[13, 853]$. After this, $k \leftarrow 2$ and *time* is updated, leading to $[x^2]_2 = [13, 853]$ and $m \leftarrow 3$. In the third iteration, applying the `fixing strategy` results in $[x^2]_3 = [13, 853, \phi]$. However, now the model becomes infeasible; $k \leftarrow 3$ and *time* is updated. Hence, we retain the previous fixed solution $[x^3]_2 \leftarrow [x^2]_2 = [13, 853]$, and the parameter M is updated to 1200 for $i = 1, 2, 3$. In the fourth iteration, the `fixing strategy` yields $[x^3]_3 \leftarrow [13, 853, \phi]$. Solving the model, we obtain the final solution $[13, 853, 1129]$. Finally,

Algorithm 3 An algorithm to solve the PP given by model (21).

Input: an instance of model (21) given by n and $M_i, \forall i = 1, \ldots, n$; time limits T and t; a scalar α; subroutines `fixing strategy` defined by Routine 1 and `branching strategy` defined by Routine 2.

Output: a feasible solution of model (21), $x_i^*, i = 1, \ldots, n$ or a statement of intractability under the given input parameters.

$k \leftarrow 0, m \leftarrow 1, time \leftarrow 0, [x^0]_0 = []$.
while $m \leq n$
    Fix $[x^k]_m \leftarrow$ `fixing strategy`$(name, [x^k]_{m-1})$.
    Solve instance of model (21) with input m and M_i for $i = 1, \ldots, m$ using `branching strategy` up to at most t seconds.
    $k \leftarrow k + 1$.
    Update *time* to wall-clock time.
    if feasible solution found
        $[x^k]_m \leftarrow$ feasible solution of model (21).
        if $m = n$
            return $[x^*]_m \leftarrow [x^k]_m$.
        $m \leftarrow m + 1$.
    else
        $[x^k]_{m-1} \leftarrow [x^{k-1}]_{m-1}$.
        $M_i \leftarrow M_i(1 + \alpha), \forall i = 1, \ldots, m$.
    if $time \geq T$
        return statement of intractability.

$k \leftarrow 4$, *time* is updated, and the algorithm terminates, returning the feasible solution $[x] = [13, 853, 1129]$.

□

Now, we compare the four fixing and five branching strategies to each other. For a given n and each of the branching strategies, we report results for each instance of the four fixing strategies; i.e., one for `Naive`, one for `SelectAll`, $n-1$ for `ExcludeOne` and $\frac{(n-1)(n-2)}{2}$ for `ExcludeTwo`. Thus, for the five branching strategies we report solutions and the time taken to obtain these solutions for $5 \cdot (2 + \frac{n(n-1)}{2})$ instances for each n; for instances that fail to obtain a feasible solution or proven infeasibility within a given time limit, we report as "Intractable". Tables S1-S6 of Appendix D present all of these results. Table 3 presents a summary of the "best" branching strategy (i.e., where a solution is obtained in the least amount of time) results for each n for each of the four fixing strategies; the corresponding best fixing strategy is always `SelectAll` (we discuss this below).

n	`fixing strategy`	`branching strategy`	Time
6	`Naive`	`Modulo` 5; 7; 11	-
	`SelectAll`	`Modulo` 1; 7; 11	-
	`ExcludeOne`	`Modulo` 1; 5; 11	-
	`ExcludeTwo`	`Modulo` 1; 11	-
7	`Naive`	`Modulo` 5	2
	`SelectAll`	`Modulo` 7; 11	-
	`ExcludeOne`	`Modulo` 7; 11	1
	`ExcludeTwo`	`Modulo` 11	1
8	`Naive`	`Modulo` 5	56
	`SelectAll`	`Modulo` 1; 7	1
	`ExcludeOne`	`Modulo` 7	8
	`ExcludeTwo`	`Modulo` 1	30
9	`Naive`	×	3000
	`SelectAll`	`Modulo` 1; 7; 11	2
	`ExcludeOne`	`Modulo` 7	11
	`ExcludeTwo`	`Modulo` 11	16
10	`Naive`	×	6000
	`SelectAll`	`Modulo` 1	18
	`ExcludeOne`	`Modulo` 1	1147
	`ExcludeTwo`	×	6000
11	`Naive`	×	6000
	`SelectAll`	`Modulo` 7	664
	`ExcludeOne`	×	6000
	`ExcludeTwo`	×	6000

Table 3: Selected results on solving model (21) with Algorithm 3 that obtain feasible solutions in the least computational time for each `fixing strategy` in Routine 2 for $n = 6$ to 11; i.e., the best performers for each fixing strategy. The Time column shows the computational time in seconds. Entries of "-" denote the solution is obtained practically instantaneously while entries of × denote no solution is obtained in the given time limit. The `Naive` branching strategy is consistently worse than the `Modulo` branching strategy. The `SelectAll` fixing strategy is consistently the best performer among the fixing strategies for any given branching strategies.

Among the fixing strategies, the `SelectAll` strategy stands out as the most effective by successfully obtaining a feasible solution in each of the 30 instances, for any branching strategy, and all $n = 5, \ldots, 10$; see, Table 4b. On the other hand, the `ExcludeOne` and `ExcludeTwo` strategies encounter a significant number of intractable instances — nearly two-fifths and three-fifths of the instances for `ExcludeOne` and `ExcludeTwo` are intractable (see, Table 4b). Both these strategies are also poor at scaling with increasing n: the proportion of intractable instances increases from

	Instances	Feasible	Avg. Time
`Naive`	167	33.5%	352
`Modulo 1`	167	53.9 %	290
`Modulo 5`	167	47.9 %	442
`Modulo 7`	167	52.1 %	377
`Modulo 11`	167	50.3 %	274

(a) `branching strategy`

	Instances	Feasible	Avg. Time
`Naive`	30	63.3 %	374
`SelectAll`	30	100.0 %	30
`ExcludeOne`	195	59.5 %	261
`ExcludeTwo`	580	40.0 %	393

(b) `fixing strategy`

Table 4: Summary of computational results on solving model (21) with Algorithm 3 for $n = 5, \ldots, 10$ for each (a) `branching strategy` and (b) `fixing strategy`. Here, Instances denotes the number of instances, Feasible denotes the percentage of instances that obtain a feasible solution in the given time limit, and Avg. Time denotes the average computational time of these feasible instances. For details, see Section 4.4 and Appendix D.

3% and 4% for $n = 7$ to 93% and 100% for $n = 10$ for `ExcludeOne` and `ExcludeTwo`, respectively.

Further, the `SelectAll` strategy consistently produces the best solutions among the four fixing strategies in 29 of its 30 instances. The only exception occurs for the case of $n = 6$, where the `Naive` and `ExcludeOne` fixing strategies obtain a solution instantly for the `Modulo` with arg $= 5$ branching strategy; however, here too the `SelectAll` strategy is worse by only one second. In contrast, the `ExcludeOne` and `ExcludeTwo` strategies yield the best solutions in a limited number of instances. Specifically, for $n = 5, \ldots, 10$, `ExcludeOne` produces the best solutions in 20/20, 3/25, 0/30, 0/35, 0/40, and 0/45 instances, respectively, while `ExcludeTwo` produces the best solutions in 30/30, 6/50, 1/75, 0/105, 0/140, and 0/180 instances, respectively (see, Tables S1-S6 of Appendix D). For $n = 5$, all instances are solved practically instantly.

Among the branching strategies, the `Modulo` with arg $= 7$ and 11 strategies both achieve the best performance in 9/24 instances; see, Table 3. This is followed by the `Modulo` with arg $= 1$ and `Modulo` with arg $= 5$ strategies which perform the best in 8/24 and 4/24 instances, respectively. In contrast, none of the instances with the `Naive` strategy achieve the lowest computation time for any n; see, Table 3. Among the `Modulo` strategies, for $n = 6, \ldots, 9$, there is no clear consensus for the best performer. For $n = 10$ and $n = 11$, five instances are intractable for any branching strategy; however, the `Modulo` with arg $= 1$ and 7 strategies consistently dominate the performance for these n.

Table 5 presents the averaged computational times, separately for each n, across instances where a feasible solution is obtained by Algorithm 3 (i.e., not intractable). We find that the `Modulo` strategy—with any of the four arguments—is almost an order of magnitude faster, on average, than the `Naive` strategy for small n. For example, consider $n = 5, 6, 7$: the slowest of the four arguments of the `Modulo` strategy (with any argument) is faster than the `Naive` in $12/12, 17/17, 16/23$ instances (there are no intractable instances in this case). It might appear that the `Naive` strategy is better for large n due to the lower computational times in Table 5. However, this masks the fact that there are only few instances where a feasible solution is obtained by the `Naive` strategy; see, Table 3. Indeed, for $n = 8, 9, 10$, the best `Modulo` strategy is faster than the `Naive` strategy in 27/30, 38/38, 47/47 of the instances, respectively. The `Naive` strategy manages to obtain feasible solutions in only 23/30, 1/38, and 1/47 instances for $n = 8$, 9, and 10, respectively; i.e., most of the nearly one-third (see, Table 4) feasible instances of the Naive strategy are from small values of n. With this background, the `Modulo` strategy outperforms the `Naive` strategy.

	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
`Naive`	127	779	356	113	156
`Modulo 1`	5	65	571	737	1186
`Modulo 5`	4	119	1082	1386	299
`Modulo 7`	5	158	857	827	221
`Modulo 11`	2	112	656	1596	1343

Table 5: Average computation time of the feasible instances alone on solving model (21) with Algorithm 3 for $n = 6, \ldots, 10$ for each `branching strategy`. Solutions for $n = 5$ are obtained practically instantaneously.

Next, we compare the four `Modulo` strategies to each other. Here, there is no clear consensus for the best performer. To see this, first consider all instances for $n = 5, \ldots, 10$. Then, the `Modulo` with arg $= 1$ strategy reports the fewest intractable instances: 77/167. In contrast, the `Modulo` with arg $= 5$, 7, and 11 strategies report higher numbers of intractable solutions: 87/167, 80/167, and 83/167, respectively. For $n = 8$, only 4/30 instances are intractable for `Modulo` with arg $= 1$; while the next best performer, `Modulo` with arg $= 7$, has 6/30 intractable instances. We find that arg $= 5$ performs relatively poorly. It takes the most computational effort (among the four `Modulo` strategies): for $n = 8, 9, 10$, its average computational time (including the time taken for intractable instances) is the worst among the four `Modulo` strategies. Further, arg $= 5$ does not outperform the other three strategies for any instance for $n = 8, 9, 10$. This pattern is particularly visible for $n = 10$ where most of the instances are intractable: the best instances of both `Modulo` with arg $= 1$ and arg $= 11$ obtain feasible solutions in under half a minute while the best for arg $= 5$ and arg $= 7$ take about four to five minutes (see, Table S6 in Appendix D). However, the under-performing `Modulo` with arg $= 5$ strategy becomes useful for $n = 11$: arg $= 5$ and arg $= 7$ are the only two strategies that obtain a feasible solution for $n = 11$ (results not shown). These varied results

provide empirical evidence in support of having multiple strategies as heuristics to solve PPs.

5. Conclusions

Employing ideas from number theory and mathematical optimization, we introduce a novel conceptual framework of optimizing over prime numbers. We describe mathematical programming formulations for several well-known problems in number theory that are traditionally studied using disciplines other than optimization. The framework we propose then facilitates numerical solutions by easily employing commercially available solvers; we do so by extending the branch-and-bound algorithm of integer programming into prime programming. Such an algorithm is generic and capable of solving any linear prime program.

We then further extend the branching rules of this method by developing additional strategies for variables to branch on as well as fixing certain variables. Our computational results, conducted on a classical conjecture to find a sequence of primes where the average of any two numbers is also a prime, shows the advantage of such methods. Future work could explore priority-based branching methods for optimization problems with integer and prime number constraints. In such problems, prioritizing branching on prime numbers could be advantageous. This preference stems from the fact that when a feasible solution satisfies the prime number restrictions, the branching process results in a larger reduction of the feasible region compared to that by branching on integer variables alone. Finally, by employing the notion of an inference dual, we extend the concept of sensitivity analysis which is well-studied in linear programming into that for prime programming.

All our data, models, and code are available at: https://github.com/montreeklim/PrimeNumberProgramming.

CRediT authorship contribution statement

Montree Jaidee: Conceptualization, Methodology, Software, Writing - Original Draft, Review & Editing. **Bismark Singh**: Conceptualization, Methodology, Writing - Original Draft, Review & Editing, Supervision.

Acknowledgement

Montree Jaidee is supported by a scholarship from the Development and Promotion of Science and Technology Talents Project (DPST). Both authors acknowledge the use of the IRIDIS High-Performance Computing Facility at the University of Southampton in completing this work.

References

Balog, A. (1992). Linear equations in primes. *Mathematika*, *39*, 367–378. doi:10.1112/S0025579300015096.

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Integer programming*. Springer International Publishing.

Dass, P., Sharma, H., Bansal, J. C., & Nygard, K. E. (2013). Meta heuristics for prime factorization problem. In *2013 World Congress on Nature and Biologically Inspired Computing* (pp. 126–131). IEEE. doi:10.1109/nabic.2013.6617850.

Dawande, M. W., & Hooker, J. N. (2000). Inference-based sensitivity analysis for mixed integer/linear programming. *Oper. Res.*, *48*, 623–634. doi:10.1287/opre.48.4.623.12420.

Granville, A. (1990). A note on sums of primes. *Can. Math. Bulletin*, *33*, 452–454. doi:10.4153/CMB-1990-073-7.

Gurobi Optimization, LLC (2024). Gurobi optimizer reference manual. `https://www.gurobi.com/documentation/current/refman/ub.html`.

Hardy, G. H., & Littlewood, J. E. (1923). Some problems of 'Partitio numerorum'; III: On the expression of a number as a sum of primes. *Acta Math-djursholm.*, *44*, 1–70. doi:10.1007/BF02403921.

Knezevic, K. (2021). Generating prime numbers using genetic algorithms. In *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)* (pp. 1224–1229). IEEE. doi:10.23919/mipro52101.2021.9597026.

Kumar, N. (2022). An alternative computational optimization technique to solve linear and non-linear Diophantine equations using discrete WQPSO algorithm. *Soft Comput.*, *26*, 12531–12544. doi:10.1007/s00500-022-07199-1.

Mangasarian, O., & Ferris, M. C. (2010). Uniqueness of integer solution of linear equations. *Optimization Letters*, *4*, 559–565. doi:10.1007/s11590-010-0183-0.

Mehrotra, H., & Pal, S. K. (2018). Using chaos in grey wolf optimizer and application to prime factorization. In *International Conference on Soft Computing for Problem Solving*. doi:10.1007/978-981-13-1592-3_3.

Pomerance, C., Sárközy, A., & Stewart, C. (1988). On divisors of sums of integers. III. *Pac. J. Math.*, *133*. doi:10.2140/pjm.1988.133.363.

Sierpiński, W. (1970). *250 problems in elementary number theory*. Modern analytic and computational methods in science and mathematics. American Elsevier Publishing Company.

Sylvester, J. J. (1881). On Tchebycheff's theory of the totality of the prime numbers comprised within given limits. *American Journal of Mathematics*, *4*, 230. doi:10.2307/2369154.

Wells, D. G. (2005). *Prime numbers: The most mysterious figures in math*. Hoboken, NJ: Wiley Hoboken.

Wolsey, L. A. (1998). *Integer programming*. New York: Wiley.

Appendix for: The Prime Programming Problem: Formulations and Solution Methods

Montree Jaidee[a], Bismark Singh[a,*]

[a] *School of Mathematical Sciences, University of Southampton, Southampton, SO17 1BJ, UK*

There are four components in this appendix. Section A provides examples of PPs which are inspired by conjectures in number theory. Section B provides the proof for Theorem 1 of the main text, required for conducting sensitivity analysis for PPs. We then provide additional examples of sensitivity analysis for PPs inspired by the Goldbach conjecture in Section C. Section D provides additional computational results to supplement those in Section 4.4 of the main text.

A. Prime Programming Formulations for Three Toy-problems

The following three problems, which can be formulated as PPs, are inspired by questions found on the websites cited below. This section provides prime programming formulations of such problems.

Definition 1. *A twin prime is a prime number that is either two more or two less than another prime number.*

Problem 1. *Find all consecutive twin prime pairs* (x_1, x_1+2) *and* (x_2, x_2+2) *such that* x_2 *is smaller than the sum of the first pair.*

Solution. The corresponding optimization model is:

$$\min_x 0 \text{ s.t.}\{x_2 \geq x_1 + 4, x_2 \leq 2x_1 + 1, x_3 = x_1 + 2, x_4 = x_2 + 2 \text{ with } x_1, x_2, x_3, x_4 \in \mathbb{P}\}. \quad (1)$$

To ensure the smallest solution pair, a variation of the problem is the following model:

$$\min_x x_1 \text{ s.t.}\{x_2 \geq x_1 + 4, x_2 \leq 2x_1 + 1 \text{ with } x_1, x_1 + 2, x_2, x_2 + 2 \in \mathbb{P}\}. \quad (2)$$

The solution to this model provides one feasible pair, (5, 7) and (11, 13); i.e., $(x_1^*, x_2^*, x_3^*, x_4^*) = (5, 11, 7, 13)$. For the next solution, we add a cutting plane $x_1 \geq x_1^* + 1$ and continue this process. Repeating this process provides us several such solutions but not necessarily *all* the solutions. In the

*Corresponding author
Email address: b.singh@southampton.ac.uk (Bismark Singh)

first five iterations of the naive branch-and-bound procedure we describe in Section 3.1 of the main text, we obtain solutions $(5, 11, 7, 13), (11, 17, 13, 19), (17, 29, 19, 31), (29, 41, 31, 43), (41, 59, 43, 61)$. However, the solutions obtained from this model are not necessarily consecutive twin prime pairs. We can verify the consecutive nature of the obtained twin primes, $(x_1^*, x_1^* + 2)$ and $(x_2^*, x_2^* + 2)$, if there exists $(x_0, x_0 + 2) \in \mathbb{P}$ for some $x_0 \leq x_2^*$. □

For a similar problem to Problem 1, see https://math.stackexchange.com/q/4014554.

Definition 2. *A super prime square is a prime number, x, such that $(a_1 + a_2 + a_3 + a_4)/6$ is prime for some $a_1, \ldots, a_4 \in \mathbb{Z}^+$ and $x + a_i$ is also a prime number for all $i = 1, \ldots, 4$.*

Problem 2. *Find the largest super prime square.*

Solution. The corresponding optimization model is:

$$\max_{x,a} x_0 \text{ s.t.}\{6x_0 = a_1 + a_2 + a_3 + a_4, x_1 = x_0 + a_1, x_2 = x_0 + a_2, x_3 = x_0 + a_3, x_4 = x_0 + a_4, \\ x_2 \geq x_1 + 1, x_3 \geq x_2 + 1, x_4 \geq x_3 + 1 \text{ with } x_0, x_1, \ldots, x_4 \in \mathbb{P} \text{ and } a_1, \ldots, a_4 \in \mathbb{Z}^+\} \quad (3)$$

To ensure the model is bounded, we need an upper bound for the variables. When the upper bound is set to 10^5, the solution to this model (using the branch-and-bound method we present in Section 3.1) provides one feasible tuple, $(x_0^*, x_1^*, x_2^*, x_3^*, x_4^*, a_1^*, a_2^*, a_3^*, a_4^*) = (3947, 9973, 9967, 9781, 9749, 6026, 6020, 5834, 5802)$ with the solution to the problem as $x_0^* = 3947$. Similarly, we obtain $x_0^* = 39727$ and $x_0^* = 399853$ when the upper bounds are set to 10^6 and 10^7, respectively. It is possible that there are infinitely many solutions. If that is the case, there is no largest super prime square. □

For a similar problem to Problem 2, see https://math.stackexchange.com/q/2515549.

Definition 3. *A sexy prime pair is a pair of prime numbers differing by 6.*

Problem 3. *Find the largest sexy prime pair $(x, x + 6)$, such that their sum is divisible by 10.*

Solution. The corresponding optimization model is:

$$\max_{x,z} x_1 \text{ s.t.}\{2x_1 + 6 = 10z, x_2 = x_1 + 6 \text{ with } x_1, x_2 \in \mathbb{P} \text{ and } z \in \mathbb{Z}^+\}. \quad (4)$$

Again, an upper bound for variables is needed to ensure the model is bounded. To simplify the model, we observe that if $2x + 6 \equiv 0 \mod 10$, then $x \equiv 2, 7 \mod 10$. Since x and $x + 6$ are primes, x must be of the form $x = 10a + 7$ for some $a \in \mathbb{Z}^+$. The simplified model is: $\max_{x,a} x_1$ s.t.$\{x_1 = 10a + 7, x_2 = x_1 + 6$ with $x_1, x_2 \in \mathbb{P}$ and $a \in \mathbb{Z}^+\}$. When the upper bound is set to 10^5, the solution to this model provides one feasible tuple, $(x_1^*, x_2^*, a^*) = (99817, 99823, 9981)$ with the sexy prime pair $(99817, 99823)$. Similarly, we obtain a sexy prime pair $(998737, 998743)$ and $(9999937, 9999943)$

when the upper bound is set to 10^6 and 10^7, respectively. It is possible that there are infinitely many solutions. If that is the case, there is no largest such sexy prime pair. □

For a similar problem to Problem 3, see https://math.stackexchange.com/q/3230707.

B. Proof of Sensitivity Analysis for PP

In this section we provide a proof of Theorem 1 of the main text. The proof mirrors the proof given in Dawande & Hooker (2000) with the key difference that the original work is for mixed-integer programs while ours considers prime programs. The central idea is to derive sufficient conditions that ensure $z^* - \Delta \leq z^*_\delta$. These conditions are informed by determining a set of infeasible (or, violated) inequalities, that the authors term as "surrogate inequalities", at each of the leaf nodes. To this end, we first drop both the prime number as well as the integer restrictions on the x variables of model (14). Then, the following relaxed model is solved at each node of the branch-and-bound tree.

$$\min_x \quad cx \tag{5a}$$

$$\text{s.t.} \quad Ax \geq b, \tag{5b}$$

$$Cx \geq d, \tag{5c}$$

$$0 \leq x \leq M. \tag{5d}$$

Here, $Cx \geq d$ denotes the branching cuts at each node where a variable x_j has a non-prime value, v. The left and right child nodes have branching cuts of the form $x_j \leq \underline{p}_v$ and $x_j \geq \bar{p}_v$, respectively. In constraint (5d), the bound of x variable is relaxed to between 0 and M; where we can write the model with an additional constraint $x_i \geq 2$ for all $i = 1, \ldots, n$ combined within the constraints (5c). Then, model (5) has the same structure as the relaxed model (17) employed in Dawande & Hooker (2000); hence, except for the branching rules of prime programs and mixed-integer programs, the surrogate inequalities at leaf nodes follow directly. Let (λ, μ, ν) be the non-negative dual multipliers of constraints (5b)-(5d), respectively. The following surrogate inequalities for model (5) directly result from Dawande & Hooker (2000).

(i) For infeasible leaf nodes, the surrogate inequality is $(\lambda A)x \geq \lambda b$.

(ii) For leaf nodes pruned by optimality, the surrogate inequality is $(\lambda A - c)x \geq \lambda b - \hat{z} + \Delta + \epsilon$, where $\hat{z}$ is the node's objective function value.

(iii) For leaf nodes pruned by bound, the surrogate inequality is $(\lambda A - c)x \geq \lambda b - \bar{z} + \Delta + \epsilon$, where $\bar{z}$ is the incumbent value of the objective function so far.

At node p, consider

$$x_j \in [\underline{v}_j, \bar{v}_j] \cap \mathbb{P},\ j = 1, \ldots, n, \tag{6}$$

where $\underline{v}, \bar{v}$ are prime numbers between 2 and M. We now define the following values as in Section 3.2,

$$z_p = \begin{cases} \epsilon, \\ \bar{z}_p - \Delta, \\ z_p^{UB} - \Delta, \end{cases} \qquad q^p = \begin{cases} \lambda^p A, \\ \lambda^p A - c, \\ \lambda^p A - c, \end{cases} \qquad q^p_\delta = \begin{cases} \lambda^p A_\delta, & : \text{if the node is pruned by infeasibility,} \\ \lambda^p A_\delta - c_\delta, & : \text{if the node is pruned by optimality,} \\ \lambda^p A_\delta - c_\delta, & : \text{if the node is pruned by bound.} \end{cases}$$

Then, the surrogate inequality at node p is $q^p x \geq \lambda^p b - z_p + \epsilon$. The corresponding perturbed surrogate inequality is then of the form $(q^p + q^p_\delta)x \geq \lambda^p(b + b_\delta) - z_p + \epsilon$. We now derive sufficient conditions such that the perturbed inequality remains infeasible for each leaf node.

Lemma S1. *Let $x_j \in [\underline{v}_j, \bar{v}_j] \cap \mathbb{P}$, $j = 1, \ldots, n$, and consider a correspondingly infeasible surrogate inequality $q^p x \geq \lambda^p b - z_p + \epsilon$. Then, the perturbed inequality $(q^p + q^p_\delta)x \geq \lambda^p(b + b_\delta) - z_p + \epsilon$ remains infeasible if and only if there exist $\bar{q}^p_1, \ldots, \bar{q}^p_n$ such that*

$$\sum_{j=1}^{n} \left((q^p_j + q^p_{\delta j})\underline{v}_j + \bar{q}^p_j(\bar{v}_j - \underline{v}_j)\right) < \lambda^p(b + b_\delta) - z_p + \epsilon, \tag{7a}$$

$$\bar{q}^p_j \geq q^p_j + q^p_{\delta j},\ \bar{q}_j \geq 0,\ j = 1, \ldots, n. \tag{7b}$$

Proof. This result is based on Lemma 2 of Dawande & Hooker (2000). The largest possible value of $(q^p + q^p_\delta)x$ is as given by the following quantity:

$$\sum_{j: q^p_j + q^p_{\delta j} < 0} (q^p_j + q^p_{\delta j})\underline{v}_j + \sum_{j: q^p_j + q^p_{\delta j} > 0} (q^p_j + q^p_{\delta j})\bar{v}_j. \tag{8}$$

$\Longrightarrow$ Assume that the inequality $(q^p + q^p_\delta)x \geq \lambda^p(b + b_\delta) - z_p + \epsilon$ is infeasible. Let $\bar{q}^p_j = \max\{q^p_j + q^p_{\delta j}, 0\}$ for $j = 1, \ldots, n$ which satisfies equation (7b). We have:

$$\begin{aligned} \sum_{j=1}^{n} \left((q^p_j + q^p_{\delta j})\underline{v}_j + \bar{q}^p_j(\bar{v}_j - \underline{v}_j)\right) &= \sum_{j: q^p_j + q^p_{\delta j} < 0} (q^p_j + q^p_{\delta j})\underline{v}_j + \sum_{j: q^p_j + q^p_{\delta j} > 0} (q^p_j + q^p_{\delta j})\underline{v}_j \\ &\quad + \sum_{j: q^p_j + q^p_{\delta j} > 0} (q^p_j + q^p_{\delta j})(\bar{v}_j - \underline{v}_j) \\ &= \sum_{j: q^p_j + q^p_{\delta j} < 0} (q^p_j + q^p_{\delta j})\underline{v}_j + \sum_{j: q^p_j + q^p_{\delta j} > 0} (q^p_j + q^p_{\delta j})\bar{v}_j \\ &< \lambda^p(b + b_\delta) - z_p + \epsilon, \end{aligned}$$

where the last inequality holds since the inequality assumed is infeasible, and the preceding expression is its largest term. Hence, equation (7) is satisfied.

$\Longleftarrow$ Now, assume that there exist $\bar{q}_1^p, \ldots, \bar{q}_n^p$ satisfying (7). Then,

$$\begin{aligned}
\lambda^p(b+b_\delta) - z_p + \epsilon &> \sum_{j=1}^{n}((q_j^p + q_{\delta\, j}^p)\underline{v}_j + \bar{q}_j^p(\bar{v}_j - \underline{v}_j)) \\
&\geq \sum_{j=1}^{n}(q_j^p + q_{\delta\, j}^p)\underline{v}_j + \sum_{j:q_j^p+q_{\delta\, j}^p>0}(q_j^p + q_{\delta\, j}^p)(\bar{v}_j - \underline{v}_j) \\
&= \sum_{j:q_j^p+q_{\delta\, j}^p<0}(q_j^p + q_{\delta\, j}^p)\underline{v}_j + \sum_{j:q_j^p+q_{\delta\, j}^p>0}(q_j^p + q_{\delta\, j}^p)\bar{v}_j \\
&> (q^p + q_\delta^p)x.
\end{aligned}$$

Hence, the inequality $(q^p + q_\delta^p)x \geq \lambda^p(b+b_\delta) - z_p + \epsilon$ is infeasible. □

Lemma S1 completes the proof of Theorem 1.

C. Another Example of Sensitivity Analysis for Prime Programming

In this section, we present an additional example of sensitivity analysis for the prime program, specifically in relation to the Goldbach conjecture.

Problem 4. *Find the largest prime number, x_1, satisfying the condition $2k+2 = x_1 + x_2$ for a given $k \in \mathbb{Z}^+$ where $x_2 \in \mathbb{P}$.*

The following PP formulates this problem with $k = 499$.

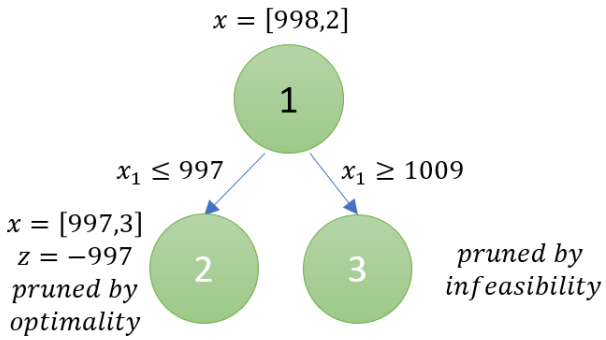
$$\begin{aligned}
z^* = \min \quad & -x_1 && \text{(9a)}\\
\text{s.t} \quad & x_1 + x_2 \geq 1000, && \text{(9b)}\\
& -x_1 - x_2 \geq -1000, && \text{(9c)}\\
& x_1, x_2 \in [2, M] \cap \mathbb{P}. && \text{(9d)}
\end{aligned}$$

We consider a perturbation of the right hand side with $b_\delta = [k_\delta, -k_\delta]$ for even k_δ. In this case, the enumeration tree of the branch-and-bound procedure described in Section 3.1 of the main text contains three nodes as in Fig. S1. For additional details, see Section 3.2 of the main text.

(i) Consider node 2. We have $\underline{x}^2 = [2,2], \bar{x}^2 = [997, M], z^2 = -997 - \Delta, \lambda^2 = [0,0]$. Then, $q^2 = [1,0], q_\delta^2 = [0,0]$. In order to satisfy the sufficient conditions of Theorem 1 at this node, we require a solution $\bar{q}_1^2, \bar{q}_2^2$ such that $2 + 995\bar{q}_1^2 + (M-2)\bar{q}_2^2 \leq 997 + \Delta$ and $\bar{q}_1^2 \geq 1, \bar{q}_2^2 \geq 0$. By setting $\bar{q}_1^2, \bar{q}_2^2 = [1,0]$ we obtain $\Delta \geq 0$.

(ii) Consider node 3. We have $\underline{x}^3 = [1009, 2], \bar{x}^3 = [M, M], z^3 = \varepsilon, \lambda^3 = [0,1]$. Then, $q^3 = [-1,-1], q_\delta^3 = [0,0]$. In order to satisfy the sufficient conditions of Theorem 1 at this node,

Figure S1: Enumeration tree for the prime branch-and-bound procedure applied to model (9). The two nodes marked in white are leaf nodes that are pruned below for reasons indicated, while the other node is further branched on. For details, see Section 3.1.



we require a solution $\vec{q}_1^3, \vec{q}_2^3$ such that $-1011 + (M - 1009)\vec{q}_1^3 + (M - 2)\vec{q}_2^3 \leq -1000 - k_\delta - \varepsilon$ and $\vec{q}_1^3, \vec{q}_2^3 \geq 0$. By setting $\vec{q}_1^3, \vec{q}_2^3 = [0, 0]$ we obtain $k_\delta \leq 11 - \varepsilon$.

The optimal solution of model (9) is $x^* = [997, 3]$. Thus, we conclude that $z_\delta^* \geq -997 - \Delta$ holds for all $\Delta \geq 0$ if $k_\delta = \{2, 4, 6, 8, 10\}$; i.e., the optimal value of x_1^* for the perturbation of model (9) does not exceed 997.

Unlike Problem 7 of the main text, enumerating all the possibilities requires checking all prime numbers below 997 (there are 167 such numbers). However, consider $k_\delta = 2$ and the corresponding optimization model: $\max_x x_1$ s.t.$\{x_1 + x_2 = 1002$ with $x_1, x_2 \in \mathbb{P}\}$. From our results of the sensitivity analysis above, we begin directly with $x_1 = 997$. Then, $[x_1, x_2] = [997, 5]$ is feasible, and, hence optimal. Similarly, for $k_\delta = 4, 8, 10$ we obtain the optimal solution in the first computation itself. For $k_\delta = 6$ the corresponding optimization model, $\max_x x_1$ s.t.$\{x_1 + x_2 = 1006$ with $x_1, x_2 \in \mathbb{P}\}$, is infeasible for $x_1 = 997$. In the second computation, we test $x_1 = 991$; i.e., the prime immediately smaller than 997. The problem is still infeasible. However, in the third iteration, we have $x_1 = 983$ and obtain an solution $[x_1, x_2] = [983, 23]$. Thus, we require at most three additional computations and there is no need to solve another optimization model. In contrast, the branch-and-bound tree for this instance of model (9) has 11 nodes.

D. Numerical Results for Linear Equations in Primes

Example S1. *An illustration of the benefit of increasing the allowed time limit for solving model* (21).

Consider an instance of model (21) solved with Routine 2 with $n = 8$, $M = 3000$ and a time limit of 300 seconds. With both the `Naive` strategy and the `Modulo` strategy with arg = 11, the instance neither obtains a feasible solution nor proves infeasibility within the considered

time limit. However, increasing the time limit to xxx and 600 seconds, respectively, provides feasible solutions. Feasible solutions are obtained relatively quickly for the `Modulo` strategy with arg $= 1, 5, 7$.

□

Example S2. *An illustration of the computational benefit from using the three enhanced variable fixing strategies of Routine 1 over the* `Naive` *fixing strategy.*

Consider an instance of model (21) with $n = 8$ and $M = 3000$ solved using the `Naive branching strategy` . The `Naive` fixing strategy takes 821 seconds to solve this instance (see Table 1). Now, consider the known solution, $\bar{x}$, for $n = 7$ with $\bar{x} = [5, 17, 41, 101, 257, 461, 521]$ as an input for Routine 2.

(i) In the `SelectAll` strategy, the first seven components are fixed; i.e., $x = \bar{x}$. Here, the instance's infeasibility is determined practically instantly. However, increasing M to 10000 provides a feasible solution in practically no extra time: $[5, 17, 41, 101, 257, 461, 521, 4157]$. This reduces the computational time from 821 seconds to only 82 seconds; the latter is the computational time to obtain the $n = 7$ solution (see Table 1).

(ii) In the `ExcludeOne` strategy with $l = 6$, we have six elements fixed. Then, we have $x = [5, 17, 41, 101, 257, 521, \phi, \phi]$. A feasible solution is determined in no time: $[5, 17, 41, 101, 257, 521, 761, 881]$. This again reduces the computational time to 82 seconds, which is the time required for the $n = 7$ solution.

(iii) In the `ExcludeTwo` strategy with $(l_1, l_2) = (6, 7)$, we have 5 elements fixed. Then, we have $\bar{x} = [5, 17, 41, 101, 257]$. A feasible solution is obtained in only an additional second: $x = [5, 17, 41, 101, 257, 521, 761, 881]$. The computational time is again reduced from 821 seconds of a `Naive` strategy (see Table 1) to 83 seconds; this includes 82 seconds to obtain the $n = 7$ solution.

□

Example S3. *An illustration of the computational benefit from using the* `Modulo` *branching strategy of Routine 2 over the* `Naive` *branching strategy.*

Consider an instance of model (21) with $n = 8$ and $M = 3000$ solved using the `Naive fixing strategy` . The `Naive` branching strategy takes 821 seconds to solve this instance (see Table 1).

(i) With the `Modulo` strategy and arg $= 1$, the instance is proven infeasible in 88 seconds. However, increasing M to 10000 provides a different feasible solution, $[13, 73, 181, 241, 373, 1693, 1801, 1861]$, in only two seconds.

(ii) With the `Modulo` strategy and arg $= 5$, the same feasible solution as the `Naive` strategy is found in 37 seconds.

(iii) With the `Modulo` strategy and arg $= 7$, the instance is proven infeasible in 246 seconds. However, increasing M to 10000 provides another feasible solution, $[7, 19, 67, 139, 607, 859, 907, 1699]$, in only two seconds.

(iv) With the `Modulo` strategy and arg $= 11$, the instance is proven infeasible in 195 seconds. However, increasing M to 10000 provides another feasible solution, $[11, 23, 71, 191, 1103, 1871, 2543, 7043]$, in 144 seconds.

□

The following tables present the numerical results obtained from Algorithm 3 using different fixing and branching strategies, each with an appropriate time limit for different values of n. All models yield solutions practically instantaneously for $n = 5$. The tables depict the computational time for each strategy for $n = 6$ to 10; the row with the lowest computation time for each branching strategy is highlighted in bold. For instances that cannot obtain a solution or report infeasibility within a given time limit, we report as "Intractable". The results of the algorithm with the inputs $n = 11$; $M_i = 1000$ for $i = 1, \ldots, n$; $T = 6000$; $t = 3000$; $\alpha = 0.2$ are not shown here since all except two cases are intractable. These two cases are the `SelectAll` fixing strategy with the `Modulo` branching strategy arguments of 5 and 7 respectively; they yield solutions $[5, 17, 881, 1181, 1637, 14957, 24197, 131297, 184721, 6353021, 69739337]$ and $[7, 19, 907, 1699, 1747, 3967, 4759, 16519, 23167, 4891507, 13542967]$ with a computation time of 3772 and 664 seconds, respectively. Our algorithm is unable to find a solution or report infeasibility in 6000 seconds for $n = 12$. For details, see Section 4.4 of the main text.

Table S1: Computational results for solving model (21) using the fixing strategies in Routine 1 and branching strategies in Routine 2. Inputs in Algorithm 3 are $n = 5$; $M_i = 1000$ for $i = 1, \ldots, n$; $T = 600$; $t = 300$; $\alpha = 0.2$. All instances obtain a solution immediately.

Strategy				**Solution**
Branching		**Fixing**		
name	**arg**	**name**	**arg**	
`Naive`	ϕ	`Naive`	ϕ	[3, 7, 19, 139, 859]
		`SelectAll`	ϕ	[3, 7, 919, 2179, 4159]
		`ExcludeOne`	[1]	[67, 127, 607, 631, 1867]
			[2]	[3, 1123, 1483, 2383, 4219]
			[3]	[3, 7, 859, 1279, 4219]
			[4]	[3, 7, 919, 2179, 4159]
		`ExcludeTwo`	[1, 2]	[1579, 1987, 1999, 2287, 4639]
			[1, 3]	[43, 823, 883, 1051, 1423]
			[1, 4]	[67, 127, 151, 331, 547]
			[2, 3]	[3, 139, 223, 523, 619]
			[2, 4]	[3, 1123, 1279, 1303, 2203]
			[3, 4]	[3, 7, 19, 139, 859]

Table S1 – *Continued from previous page*

Strategy				Solution
Branching		**Fixing**		
name	**arg**	**name**	**arg**	
`Modulo`	1	`Naive`	ϕ	[13, 61, 73, 241, 853]
		`SelectAll`	ϕ	[13, 853, 1129, 1213, 2113]
		`ExcludeOne`	[1]	[1321, 1621, 1873, 3001, 4813]
			[2]	[13, 1129, 1213, 2113, 2389]
			[3]	[13, 853, 1213, 1753, 2113]
			[4]	[13, 853, 1129, 1213, 2389]
		`ExcludeTwo`	[1, 2]	[241, 373, 601, 853, 1381]
			[1, 3]	[1237, 1621, 1861, 3457, 4597]
			[1, 4]	[1321, 1621, 1777, 3001, 3457]
			[2, 3]	[13, 241, 373, 601, 853]
			[2, 4]	[13, 1129, 1213, 2113, 2389]
			[3, 4]	[13, 853, 1453, 1489, 1609]
`Modulo`	5	`Naive`	ϕ	[5, 17, 29, 89, 449]
		`SelectAll`	ϕ	[5, 17, 881, 1181, 1637]
		`ExcludeOne`	[1]	[1229, 1637, 2777, 2897, 2909]
			[2]	[5, 53, 89, 113, 929]
			[3]	[5, 17, 509, 617, 1109]
			[4]	[5, 17, 881, 1181, 1637]
		`ExcludeTwo`	[1, 2]	[89, 113, 389, 449, 1433]
			[1, 3]	[1361, 1637, 2417, 3257, 3461]
			[1, 4]	[1229, 1637, 2417, 2897, 2909]
			[2, 3]	[5, 89, 113, 269, 929]
			[2, 4]	[5, 53, 89, 929, 953]
			[3, 4]	[5, 17, 29, 89, 449]
`Modulo`	7	`Naive`	ϕ	[7, 19, 67, 127, 607]
		`SelectAll`	ϕ	[7, 19, 907, 1699, 1747]
		`ExcludeOne`	[1]	[1087, 2011, 2467, 3691, 4987]
			[2]	[7, 967, 1279, 3307, 3727]
			[3]	[7, 19, 907, 1699, 1747]
			[4]	[7, 19, 907, 1699, 1747]
		`ExcludeTwo`	[1, 2]	[1231, 1327, 1531, 1567, 1987]
			[1, 3]	[1447, 1867, 1879, 3499, 3559]
			[1, 4]	[1087, 2011, 2251, 2311, 3691]
			[2, 3]	[7, 127, 151, 271, 547]
			[2, 4]	[7, 967, 1531, 2131, 3571]
			[3, 4]	[7, 19, 67, 859, 907]
`Modulo`	11	`Naive`	ϕ	[11, 23, 71, 191, 443]
		`SelectAll`	ϕ	[11, 23, 911, 1283, 2711]
		`ExcludeOne`	[1]	[1223, 1979, 2243, 2939, 3083]
			[2]	[11, 1031, 1187, 1931, 2207]
			[3]	[11, 23, 911, 1283, 2843]

Table S1 – *Continued from previous page*

Strategy				Solution
Branching		**Fixing**		
name	**arg**	**name**	**arg**	
			[4]	[11, 23, 911, 1283, 2843]
		`ExcludeTwo`	[1, 2]	[191, 827, 947, 2027, 2447]
			[1, 3]	[1223, 2243, 2423, 2999, 3083]
			[1, 4]	[1223, 1979, 2243, 2939, 3083]
			[2, 3]	[11, 911, 947, 1607, 1667]
			[2, 4]	[11, 1031, 1163, 1583, 2111]
			[3, 4]	[11, 23, 71, 191, 443]

Table S2: Computational results for solving model (21) using the fixing strategies in Routine 1 and the branching strategies in Routine 2. Inputs in Algorithm 3 are $n = 6$; $M_i = 1000$ for $i = 1, \ldots, n$; $T = 3000$; $t = 600$; $\alpha = 0.2$.

Strategy				Solution	Time (s)
Branching		Fixing			
name	arg	name	arg		
`Naive`	ϕ	`Naive`	ϕ	[3, 11, 23, 71, 191, 443]	120
		`SelectAll`	ϕ	[3, 7, 919, 2179, 4159, 13399]	**1**
		`ExcludeOne`	[1]	[127, 607, 631, 1867, 8971, 10267]	21
			[2]	[3, 1483, 2383, 4219, 6823, 13723]	18
			[3]	[3, 7, 1279, 4219, 7159, 8179]	13
			[4]	[3, 7, 919, 4159, 13399, 37039]	255
			[5]	[3, 7, 919, 2179, 4159, 13399]	36
		`ExcludeTwo`	[1, 2]	[1999, 2287, 4639, 6199, 9007, 10459]	74
			[1, 3]	[823, 1051, 1423, 2083, 2143, 4951]	22
			[1, 4]	[127, 151, 547, 607, 1951, 2791]	22
			[1, 5]	[127, 607, 631, 3931, 6007, 8971]	443
			[2, 3]	[3, 523, 619, 859, 1723, 2719]	16
			[2, 4]	[3, 1279, 2203, 3163, 3463, 4423]	15
			[2, 5]	[3, 1483, 2383, 2551, 4051, 6163]	13
			[3, 4]	[3, 7, 859, 1279, 4219, 8179]	149
			[3, 5]	[3, 7, 1279, 4759, 5179, 8179]	228
			[4, 5]	[3, 7, 919, 2179, 4159, 13399]	716
`Modulo`	1	`Naive`	ϕ	[13, 61, 73, 241, 601, 853]	2
		`SelectAll`	ϕ	[13, 853, 1129, 1213, 2113, 2389]	**0**
		`ExcludeOne`	[1]	[1621, 1873, 3001, 4813, 5521, 21313]	8
			[2]	[13, 1213, 2113, 2389, 9013, 15493]	9
			[3]	[13, 853, 1753, 2113, 3121, 4021]	1
			[4]	[13, 853, 1129, 2389, 3049, 3313]	1
			[5]	[13, 853, 1129, 1213, 2113, 2389]	**0**
		`ExcludeTwo`	[1, 2]	[601, 853, 1381, 2341, 2593, 3121]	2
			[1, 3]	[1621, 3457, 4597, 5641, 5857, 7417]	7
			[1, 4]	[1621, 1777, 3457, 4597, 5641, 7417]	5
			[1, 5]	[1621, 1873, 3001, 9433, 10141, 11593]	39
			[2, 3]	[13, 601, 853, 1213, 3373, 4021]	5
			[2, 4]	[13, 1213, 2389, 2953, 3049, 3733]	1
			[2, 5]	[13, 1213, 2113, 2293, 2389, 7333]	10
			[3, 4]	[13, 853, 1609, 2053, 2113, 2389]	1
			[3, 5]	[13, 853, 1753, 2113, 3121, 4021]	2
			[4, 5]	[13, 853, 1129, 1213, 2113, 2389]	**0**
`Modulo`	5	`Naive`	ϕ	[5, 17, 41, 101, 257, 461]	**0**
		`SelectAll`	ϕ	[5, 17, 881, 1181, 1637, 14957]	1
		`ExcludeOne`	[1]	[1637, 2777, 2897, 2909, 3917, 6389]	2
			[2]	[5, 89, 113, 929, 1193, 1949]	**0**
			[3]	[5, 17, 617, 1109, 1277, 1709]	**0**
			[4]	[5, 17, 881, 1637, 2357, 4877]	1

Table S2 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[5]	[5, 17, 881, 1181, 9281, 11717]	7
		ExcludeTwo	[1, 2]	[389, 449, 1433, 3413, 3833, 5009]	4
			[1, 3]	[1637, 3257, 3461, 4637, 5441, 10457]	12
			[1, 4]	[1637, 2417, 2909, 3917, 6257, 6869]	7
			[1, 5]	[1637, 2777, 2897, 3821, 4001, 4157]	1
			[2, 3]	[5, 269, 929, 1277, 1949, 2897]	1
			[2, 4]	[5, 89, 953, 1109, 4133, 6653]	12
			[2, 5]	[5, 89, 113, 389, 449, 1433]	1
			[3, 4]	[5, 17, 449, 509, 1877, 4817]	6
			[3, 5]	[5, 17, 617, 1109, 1277, 1709]	1
			[4, 5]	[5, 17, 881, 1301, 2861, 3797]	4
Modulo	7	Naive	ϕ	[7, 19, 67, 127, 547, 607]	**0**
		SelectAll	ϕ	[7, 19, 907, 1699, 1747, 3967]	**0**
		ExcludeOne	[1]	[2011, 2467, 3691, 4987, 13567, 28771]	21
			[2]	[7, 1279, 3307, 3727, 4159, 5407]	1
			[3]	[7, 19, 1699, 1747, 3967, 4759]	1
			[4]	[7, 19, 907, 1747, 3967, 4759]	1
			[5]	[7, 19, 907, 1699, 1747, 3967]	1
		ExcludeTwo	[1, 2]	[1531, 1567, 1987, 2791, 3547, 3847]	2
			[1, 3]	[1867, 3499, 3559, 3727, 4759, 8179]	7
			[1, 4]	[2011, 2251, 3691, 4363, 5443, 7963]	6
			[1, 5]	[2011, 2467, 3691, 6211, 6247, 11887]	25
			[2, 3]	[7, 271, 547, 607, 967, 2791]	2
			[2, 4]	[7, 1531, 3571, 3967, 6991, 8191]	7
			[2, 5]	[7, 1279, 3307, 3727, 4159, 5407]	2
			[3, 4]	[7, 19, 907, 1699, 1747, 3967]	6
			[3, 5]	[7, 19, 1699, 2887, 3727, 3739]	3
			[4, 5]	[7, 19, 907, 1699, 1747, 3967]	6
Modulo	11	Naive	ϕ	[11, 23, 83, 251, 443, 683]	**0**
		SelectAll	ϕ	[11, 23, 911, 1283, 2711, 2843]	**0**
		ExcludeOne	[1]	[1979, 2243, 2939, 3083, 4919, 5099]	1
			[2]	[11, 1187, 1931, 2207, 5147, 5231]	1
			[3]	[11, 23, 1283, 2843, 3203, 12611]	4
			[4]	[11, 23, 911, 2843, 4691, 5903]	1
			[5]	[11, 23, 911, 1283, 2711, 2843]	**0**
		ExcludeTwo	[1, 2]	[947, 2027, 2447, 4211, 5867, 5987]	5
			[1, 3]	[2243, 2999, 3083, 5903, 6299, 7703]	6
			[1, 4]	[1979, 2243, 3083, 4679, 4919, 5099]	2
			[1, 5]	[1979, 2243, 2939, 3083, 4919, 5099]	2
			[2, 3]	[11, 1607, 1667, 2087, 2591, 3011]	1
			[2, 4]	[11, 1163, 2111, 2843, 5903, 7823]	10

Table S2 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		Fixing			
name	**arg**	**name**	**arg**		
			[2, 5]	[11, 1187, 1931, 2207, 5147, 5231]	3
			[3, 4]	[11, 23, 443, 491, 683, 911]	**0**
			[3, 5]	[11, 23, 1283, 2711, 2843, 3203]	2
			[4, 5]	[11, 23, 911, 1283, 2711, 2843]	1

Table S3: Computational results for solving model (21) using the fixing strategies in Routine 1 and the branching strategies in Routine 2. Inputs in Algorithm 3 are $n = 7$; $M_i = 1000$ for $i = 1, \ldots, n$; $T = 3000$; $t = 600$; $\alpha = 0.2$.

Strategy				Solution	Time (s)
Branching		Fixing			
name	arg	name	arg		
`Naive`	ϕ	`Naive`	ϕ	[5, 17, 41, 101, 257, 461, 521]	243
		`SelectAll`	ϕ	[3, 7, 919, 2179, 4159, 13399, 37039]	**4**
		`ExcludeOne`	[1]	[607, 631, 1867, 8971, 10267, 10687, 24151]	42
			[2]	[3, 2383, 4219, 6823, 13723, 14083, 25639]	29
			[3]	[3, 7, 4219, 7159, 8179, 10039, 20599]	36
			[4]	[3, 7, 919, 13399, 37039, 44179, 191599]	2940
			[5]	Intractable	3000
			[6]	[3, 7, 919, 2179, 4159, 13399, 37039]	167
		`ExcludeTwo`	[1, 2]	[4639, 6199, 9007, 10459, 18127, 21379, 37987]	942
			[1, 3]	[1051, 2083, 2143, 4951, 13063, 33331, 39043]	2077
			[1, 4]	[151, 547, 1951, 2791, 6091, 6451, 8167]	60
			[1, 5]	[607, 631, 3931, 8971, 10687, 10771, 11047]	455
			[1, 6]	[607, 631, 1867, 8971, 10267, 10687, 24151]	600
			[2, 3]	[3, 859, 1723, 2719, 4219, 5503, 8179]	69
			[2, 4]	[3, 2203, 3463, 4423, 14419, 21943, 25219]	1424
			[2, 5]	[3, 2383, 2551, 6163, 8263, 8311, 15091]	253
			[2, 6]	[3, 2383, 4219, 6823, 10039, 20599, 25903]	318
			[3, 4]	[3, 7, 4219, 8179, 10039, 20599, 29179]	1352
			[3, 5]	Intractable	3000
			[3, 6]	[3, 7, 4219, 7159, 8179, 10039, 20599]	422
			[4, 5]	Intractable	3000
			[4, 6]	Intractable	3000
			[5, 6]	[3, 7, 919, 2179, 4159, 37039, 42379]	3372
`Modulo`	1	`Naive`	ϕ	[73, 241, 601, 853, 1021, 1213, 1381]	367
		`SelectAll`	ϕ	[13, 853, 1129, 1213, 2113, 2389, 7333]	**1**
		`ExcludeOne`	[1]	[1873, 3001, 4813, 5521, 21313, 42433, 68053]	79
			[2]	[13, 2113, 2389, 9013, 15493, 18913, 72613]	87
			[3]	[13, 853, 2113, 3121, 4021, 9973, 21661]	11
			[4]	[13, 853, 1129, 3049, 3313, 7993, 10009]	3
			[5]	[13, 853, 1129, 1213, 2389, 3049, 20593]	10
			[6]	[13, 853, 1129, 1213, 2113, 2389, 7333]	2
		`ExcludeTwo`	[1, 2]	[1381, 2341, 2593, 3121, 8521, 9733, 12613]	22
			[1, 3]	[3457, 5641, 5857, 7417, 11821, 36637, 37321]	207
			[1, 4]	[1777, 3457, 5641, 7417, 9337, 23197, 24061]	102
			[1, 5]	[1873, 3001, 9433, 11593, 21313, 23293, 23833]	94
			[1, 6]	[1873, 3001, 4813, 5521, 5881, 7333, 21313]	12
			[2, 3]	[13, 1213, 3373, 4021, 6673, 9901, 14653]	38
			[2, 4]	[13, 2389, 3049, 3733, 9109, 18253, 20593]	86
			[2, 5]	[13, 2113, 2293, 7333, 11353, 16573, 17449]	53
			[2, 6]	[13, 2113, 2389, 9013, 19429, 21529, 36973]	143

Table S3 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[3, 4]	[13, 853, 2113, 2389, 3313, 6733, 9013]	18
			[3, 5]	[13, 853, 2113, 4021, 8713, 9973, 21661]	97
			[3, 6]	[13, 853, 2113, 3121, 3313, 9013, 11701]	23
			[4, 5]	[13, 853, 1129, 2389, 3049, 3313, 10009]	19
			[4, 6]	[13, 853, 1129, 3049, 3313, 7993, 10009]	16
			[5, 6]	[13, 853, 1129, 1213, 2113, 2389, 7333]	12
`Modulo`	5	`Naive`	ϕ	[5, 17, 41, 101, 257, 461, 521]	**2**
		`SelectAll`	ϕ	[5, 17, 881, 1181, 1637, 14957, 24197]	**2**
		`ExcludeOne`	[1]	[2777, 2897, 2909, 3917, 6389, 44357, 49937]	87
			[2]	[5, 113, 929, 1193, 1949, 8669, 29333]	28
			[3]	[5, 17, 1109, 1277, 1709, 16217, 16229]	12
			[4]	[5, 17, 881, 2357, 4877, 28097, 48821]	92
			[5]	[5, 17, 881, 1181, 11717, 14957, 24197]	12
			[6]	[5, 17, 881, 1181, 1637, 14957, 24197]	18
		`ExcludeTwo`	[1, 2]	[1433, 3413, 3833, 5009, 6329, 14249, 15149]	34
			[1, 3]	[3257, 4637, 5441, 10457, 18521, 19841, 27281]	173
			[1, 4]	[2417, 2909, 6257, 6869, 9377, 16529, 20897]	57
			[1, 5]	[2777, 2897, 3821, 4157, 19001, 24821, 54101]	1028
			[1, 6]	[2777, 2897, 2909, 3917, 6689, 12269, 23189]	77
			[2, 3]	[5, 1277, 1949, 2897, 6977, 8669, 14489]	52
			[2, 4]	[5, 953, 4133, 6653, 9833, 19433, 23549]	89
			[2, 5]	[5, 113, 389, 1433, 1709, 6269, 6329]	7
			[2, 6]	[5, 113, 929, 1193, 5309, 7589, 14969]	73
			[3, 4]	[5, 17, 1877, 4817, 13121, 14897, 20261]	93
			[3, 5]	[5, 17, 1109, 1709, 2969, 10949, 11597]	16
			[3, 6]	[5, 17, 1109, 1277, 1889, 10457, 15749]	70
			[4, 5]	[5, 17, 881, 3797, 7817, 10457, 11717]	33
			[4, 6]	[5, 17, 881, 2357, 7817, 19697, 44537]	625
			[5, 6]	[5, 17, 881, 1181, 9281, 11717, 14957]	53
`Modulo`	7	`Naive`	ϕ	[7, 19, 67, 139, 607, 859, 907]	4
		`SelectAll`	ϕ	[7, 19, 907, 1699, 1747, 3967, 4759]	**0**
		`ExcludeOne`	[1]	[2467, 3691, 4987, 13567, 28771, 45307, 75211]	69
			[2]	[7, 3307, 3727, 4159, 5407, 27127, 39727]	51
			[3]	[7, 19, 1747, 3967, 4759, 8707, 16519]	7
			[4]	[7, 19, 907, 3967, 4759, 15427, 16519]	9
			[5]	[7, 19, 907, 1699, 3967, 4759, 15427]	8
			[6]	[7, 19, 907, 1699, 1747, 3967, 4759]	1
		`ExcludeTwo`	[1, 2]	[1987, 2791, 3547, 3847, 8167, 8887, 9511]	15
			[1, 3]	[3499, 3727, 4759, 8179, 25087, 30259, 38839]	238
			[1, 4]	[2251, 3691, 5443, 7963, 10303, 25171, 31723]	87
			[1, 5]	[2467, 3691, 6211, 11887, 25147, 27091, 33391]	218

Table S3 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[1, 6]	[2467, 3691, 4987, 13567, 20407, 21211, 56731]	246
			[2, 3]	[7, 607, 967, 2791, 6451, 14407, 30871]	119
			[2, 4]	[7, 3571, 6991, 8191, 10687, 12547, 14071]	22
			[2, 5]	[7, 3307, 3727, 5407, 8179, 10039, 13399]	18
			[2, 6]	[7, 3307, 3727, 4159, 14479, 20599, 21019]	79
			[3, 4]	[7, 19, 1747, 3967, 5347, 8707, 12739]	36
			[3, 5]	[7, 19, 2887, 3739, 11467, 27739, 74719]	2214
			[3, 6]	[7, 19, 1747, 3967, 5347, 8707, 12739]	30
			[4, 5]	[7, 19, 907, 3967, 4759, 15427, 16519]	79
			[4, 6]	[7, 19, 907, 3967, 4759, 15427, 16519]	73
			[5, 6]	[7, 19, 907, 1699, 1747, 3967, 4759]	2
`Modulo`	11	`Naive`	ϕ	[23, 179, 359, 443, 599, 683, 839]	68
		`SelectAll`	ϕ	[11, 23, 911, 1283, 2711, 2843, 3803]	**0**
		`ExcludeOne`	[1]	[2243, 2939, 3083, 4919, 5099, 9323, 21563]	9
			[2]	[11, 1931, 2207, 5147, 5231, 8171, 9767]	3
			[3]	[11, 23, 2843, 3203, 12611, 83003, 91631]	208
			[4]	[11, 23, 911, 4691, 5903, 6491, 27743]	17
			[5]	[11, 23, 911, 1283, 2843, 5903, 12611]	6
			[6]	[11, 23, 911, 1283, 2711, 2843, 3803]	1
		`ExcludeTwo`	[1, 2]	[2447, 4211, 5867, 5987, 9311, 13007, 47387]	909
			[1, 3]	[2999, 5903, 6299, 7703, 20939, 24419, 25643]	182
			[1, 4]	[2243, 3083, 4919, 5099, 8219, 10739, 24419]	96
			[1, 5]	[2243, 2939, 3083, 5099, 5903, 11483, 44879]	305
			[1, 6]	[2243, 2939, 3083, 4919, 5099, 9323, 21563]	89
			[2, 3]	[11, 2087, 2591, 3011, 15107, 20147, 22787]	108
			[2, 4]	[11, 2111, 5903, 7823, 8663, 9851, 15791]	16
			[2, 5]	[11, 1931, 2207, 5231, 5387, 6491, 8171]	4
			[2, 6]	[11, 1931, 2207, 5147, 5231, 8171, 9767]	9
			[3, 4]	[11, 23, 683, 911, 1283, 2711, 3803]	3
			[3, 5]	[11, 23, 2711, 3203, 4691, 9923, 12503]	17
			[3, 6]	[11, 23, 2843, 3203, 4691, 17483, 24971]	159
			[4, 5]	[11, 23, 911, 2843, 3803, 24971, 28631]	199
			[4, 6]	[11, 23, 911, 4691, 6491, 6911, 26891]	174
			[5, 6]	[11, 23, 911, 1283, 2711, 2843, 3803]	1

Table S4: Computational results for solving model (21) using either fixing strategy in Routine 1 or branching strategy in Routine 2. Inputs in Algorithm 3 are $n = 8$; $M_i = 1000$ for $i = 1, \ldots, n$; $T = 3000$; $t = 600$; $\alpha = 0.2$.

Strategy				Solution	Time (s)
Branching		Fixing			
name	arg	name	arg		
`Naive`	ϕ	`Naive`	ϕ	Intractable	3000
		`SelectAll`	ϕ	[3, 7, 919, 2179, 4159, 13399, 37039, 301759]	**25**
		`ExcludeOne`	[1]	Intractable	3000
			[2]	[3, 4219, 6823, 13723, 14083, 25639, 25903, 40039]	44
			[3]	[3, 7, 7159, 8179, 10039, 20599, 22699, 29179]	42
			[4]	Intractable	3000
			[5]	Intractable	3000
			[6]	Intractable	3000
			[7]	Intractable	3000
		`ExcludeTwo`	[1, 2]	Intractable	3000
			[1, 3]	Intractable	3000
			[1, 4]	Intractable	3000
			[1, 5]	[631, 3931, 8971, 10771, 11047, 11827, 18691, 27367]	996
			[1, 6]	Intractable	3000
			[1, 7]	Intractable	3000
			[2, 3]	[3, 2719, 4219, 5503, 8179, 16843, 23623, 27763]	777
			[2, 4]	Intractable	3000
			[2, 5]	Intractable	3000
			[2, 6]	Intractable	3000
			[2, 7]	[3, 4219, 6823, 13723, 14083, 22543, 25639, 25903]	321
			[3, 4]	Intractable	3000
			[3, 5]	Intractable	3000
			[3, 6]	Intractable	3000
			[3, 7]	[3, 7, 7159, 8179, 10039, 20599, 22699, 29179]	286
			[4, 5]	Intractable	3000
			[4, 6]	Intractable	3000
			[4, 7]	Intractable	3000
			[5, 6]	Intractable	3000
			[5, 7]	Intractable	3000
			[6, 7]	Intractable	3000
`Modulo`	1	`Naive`	ϕ	[13, 73, 181, 241, 373, 1693, 1801, 1861]	1579
		`SelectAll`	ϕ	[13, 853, 1129, 1213, 2113, 2389, 7333, 20593]	**1**
		`ExcludeOne`	[1]	[3001, 4813, 5521, 21313, 42433, 68053, 70573, 178693]	254
			[2]	[13, 2389, 9013, 15493, 18913, 72613, 176509, 283009]	1268
			[3]	[13, 853, 3121, 4021, 9973, 21661, 46273, 118093]	259
			[4]	[13, 853, 1129, 3313, 7993, 10009, 20173, 30169]	16
			[5]	[13, 853, 1129, 1213, 3049, 20593, 53353, 128053]	237
			[6]	[13, 853, 1129, 1213, 2113, 7333, 20593, 41113]	42
			[7]	[13, 853, 1129, 1213, 2113, 2389, 7333, 20593]	11
		`ExcludeTwo`	[1, 2]	[2593, 3121, 8521, 9733, 12613, 12781, 46441, 72673]	1026

Table S4 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[1, 3]	Intractable	3000
			[1, 4]	[3457, 5641, 9337, 23197, 24061, 32797, 34381, 112921]	1757
			[1, 5]	Intractable	3000
			[1, 6]	Intractable	3000
			[1, 7]	Intractable	3000
			[2, 3]	[13, 4021, 6673, 9901, 14653, 22861, 74713, 78301]	1472
			[2, 4]	[13, 3049, 9109, 18253, 20593, 21433, 21529, 72109]	883
			[2, 5]	[13, 2293, 7333, 16573, 17449, 19009, 24469, 26053]	58
			[2, 6]	[13, 2389, 9013, 19429, 36973, 38149, 45553, 75109]	221
			[2, 7]	[13, 2389, 9013, 15493, 18913, 42373, 54949, 72613]	256
			[3, 4]	[13, 853, 3313, 6733, 9013, 18013, 37573, 38149]	376
			[3, 5]	[13, 853, 4021, 9973, 21661, 46273, 81373, 118093]	1335
			[3, 6]	[13, 853, 3121, 3313, 11701, 24061, 36973, 53281]	283
			[3, 7]	[13, 853, 3121, 4021, 9973, 46273, 57301, 79861]	1040
			[4, 5]	[13, 853, 1129, 3313, 10009, 20173, 30169, 64969]	770
			[4, 6]	[13, 853, 1129, 3313, 10009, 20173, 30169, 64969]	772
			[4, 7]	[13, 853, 1129, 3313, 7993, 8389, 10009, 20173]	30
			[5, 6]	[13, 853, 1129, 1213, 7333, 19009, 20593, 51109]	662
			[5, 7]	[13, 853, 1129, 1213, 3049, 7993, 20593, 45289]	178
			[6, 7]	[13, 853, 1129, 1213, 2113, 2389, 7333, 20593]	69
`Modulo`	5	`Naive`	ϕ	[5, 17, 41, 101, 257, 521, 761, 881]	56
		`SelectAll`	ϕ	[5, 17, 881, 1181, 1637, 14957, 24197, 131297]	**6**
		`ExcludeOne`	[1]	[2897, 2909, 3917, 6389, 44357, 49937, 63809, 65609]	90
			[2]	[5, 929, 1193, 1949, 8669, 29333, 42209, 222533]	861
			[3]	Intractable	3000
			[4]	Intractable	3000
			[5]	[5, 17, 881, 1181, 14957, 24197, 113357, 131297]	340
			[6]	[5, 17, 881, 1181, 1637, 24197, 44537, 108821]	147
			[7]	[5, 17, 881, 1181, 1637, 14957, 51341, 92357]	177
		`ExcludeTwo`	[1, 2]	[3833, 5009, 6329, 14249, 15149, 25253, 33353, 33569]	110
			[1, 3]	Intractable	3000
			[1, 4]	[2909, 6257, 9377, 16529, 20897, 22229, 23057, 57329]	487
			[1, 5]	Intractable	3000
			[1, 6]	Intractable	3000
			[1, 7]	[2897, 2909, 3917, 6389, 44357, 49937, 63809, 65609]	324
			[2, 3]	[5, 2897, 6977, 8669, 14489, 22937, 30449, 37337]	201
			[2, 4]	[5, 4133, 9833, 19433, 23549, 35969, 127709, 217733]	2771
			[2, 5]	[5, 389, 1433, 6269, 6329, 12653, 17609, 65993]	1053
			[2, 6]	[5, 929, 1193, 5309, 14969, 31769, 33773, 77489]	1046
			[2, 7]	[5, 929, 1193, 1949, 8669, 56873, 68633, 84713]	1534
			[3, 4]	[5, 17, 13121, 14897, 20261, 23957, 50321, 98057]	2828
			[3, 5]	[5, 17, 1709, 10949, 11597, 14369, 52769, 64109]	465

Table S4 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[3, 6]	Intractable	3000
			[3, 7]	Intractable	3000
			[4, 5]	Intractable	3000
			[4, 6]	[5, 17, 881, 7817, 44537, 48821, 92357, 131297]	2164
			[4, 7]	Intractable	3000
			[5, 6]	[5, 17, 881, 1181, 14957, 24197, 113357, 131297]	2870
			[5, 7]	[5, 17, 881, 1181, 14957, 24197, 113357, 131297]	2812
			[6, 7]	[5, 17, 881, 1181, 1637, 28097, 51341, 82361]	2375
`Modulo`	7	`Naive`	ϕ	[7, 19, 67, 379, 859, 907, 1699, 1747]	2052
		`SelectAll`	ϕ	[7, 19, 907, 1699, 1747, 3967, 4759, 16519]	**1**
		`ExcludeOne`	[1]	[3691, 4987, 13567, 28771, 45307, 75211, 111427, 397687]	1747
			[2]	Intractable	3000
			[3]	[7, 19, 3967, 4759, 8707, 16519, 23167, 43987]	25
			[4]	[7, 19, 907, 4759, 15427, 16519, 34747, 56767]	68
			[5]	[7, 19, 907, 1699, 4759, 15427, 16519, 34747]	17
			[6]	[7, 19, 907, 1699, 1747, 4759, 15727, 23167]	14
			[7]	[7, 19, 907, 1699, 1747, 3967, 4759, 16519]	8
		`ExcludeTwo`	[1, 2]	[3547, 3847, 8167, 8887, 9511, 34171, 34591, 105907]	2408
			[1, 3]	[3727, 8179, 25087, 30259, 38839, 39619, 52267, 64879]	328
			[1, 4]	Intractable	3000
			[1, 5]	[3691, 6211, 11887, 27091, 33391, 66271, 81727, 130687]	1651
			[1, 6]	[3691, 4987, 13567, 20407, 56731, 80347, 103567, 184711]	2043
			[1, 7]	Intractable	3000
			[2, 3]	[7, 2791, 6451, 14407, 30871, 76231, 79231, 95191]	843
			[2, 4]	[7, 6991, 10687, 12547, 14071, 16567, 98731, 155851]	2582
			[2, 5]	[7, 3727, 5407, 10039, 13399, 29587, 38839, 43627]	207
			[2, 6]	Intractable	3000
			[2, 7]	[7, 3727, 4159, 5407, 27127, 38167, 47119, 67867]	422
			[3, 4]	Intractable	3000
			[3, 5]	Intractable	3000
			[3, 6]	[7, 19, 3967, 5347, 12739, 17839, 102679, 125299]	2550
			[3, 7]	[7, 19, 3967, 4759, 8707, 16519, 23167, 43987]	187
			[4, 5]	[7, 19, 907, 15427, 16519, 32359, 56767, 101527]	2404
			[4, 6]	[7, 19, 907, 4759, 16519, 23167, 27739, 37567]	160
			[4, 7]	[7, 19, 907, 4759, 15427, 16519, 34747, 56767]	246
			[5, 6]	[7, 19, 907, 1699, 4759, 15427, 16519, 34747]	283
			[5, 7]	[7, 19, 907, 1699, 4759, 15427, 16519, 34747]	254
			[6, 7]	[7, 19, 907, 1699, 1747, 3967, 4759, 16519]	64
`Modulo`	11	`Naive`	ϕ	[11, 83, 251, 263, 443, 1103, 1511, 2111]	2605
		`SelectAll`	ϕ	[11, 23, 911, 1283, 2711, 2843, 3803, 24971]	**2**
		`ExcludeOne`	[1]	[2939, 3083, 4919, 5099, 9323, 21563, 34403, 87443]	88

Table S4 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		Fixing			
name	arg	name	arg		
			[2]	[11, 2207, 5147, 5231, 8171, 9767, 21767, 87767]	160
			[3]	Intractable	3000
			[4]	[11, 23, 911, 5903, 6491, 27743, 52511, 61331]	47
			[5]	[11, 23, 911, 1283, 5903, 12611, 49331, 91631]	179
			[6]	[11, 23, 911, 1283, 2711, 3803, 24971, 28631]	27
			[7]	[11, 23, 911, 1283, 2711, 2843, 3803, 24971]	16
		`ExcludeTwo`	[1, 2]	Intractable	3000
			[1, 3]	[5903, 7703, 20939, 24419, 25643, 33623, 69383, 79379]	542
			[1, 4]	[3083, 4919, 8219, 10739, 24419, 30539, 36599, 42719]	234
			[1, 5]	Intractable	3000
			[1, 6]	[2939, 3083, 4919, 5099, 21563, 26183, 34403, 87443]	593
			[1, 7]	[2939, 3083, 4919, 5099, 9323, 24419, 34403, 52223]	791
			[2, 3]	Intractable	3000
			[2, 4]	[11, 5903, 8663, 9851, 15791, 24203, 31583, 108863]	1306
			[2, 5]	[11, 2207, 5231, 6491, 8171, 17471, 35027, 61007]	704
			[2, 6]	[11, 2207, 5147, 5231, 9767, 20771, 21767, 87767]	2140
			[2, 7]	[11, 2207, 5147, 5231, 8171, 9767, 21767, 87767]	2281
			[3, 4]	[11, 23, 1283, 2711, 3803, 21683, 24971, 29063]	307
			[3, 5]	[11, 23, 3203, 9923, 12503, 29243, 32363, 67523]	1242
			[3, 6]	Intractable	3000
			[3, 7]	Intractable	3000
			[4, 5]	Intractable	3000
			[4, 6]	Intractable	3000
			[4, 7]	[11, 23, 911, 5903, 6491, 9851, 27743, 34763]	202
			[5, 6]	[11, 23, 911, 1283, 3803, 5783, 24971, 28631]	196
			[5, 7]	[11, 23, 911, 1283, 5903, 29243, 39971, 57383]	598
			[6, 7]	[11, 23, 911, 1283, 2711, 2843, 3803, 24971]	166

Table S5: Computational results for solving model (21) using the fixing strategies in Routine 1 and the branching strategies in Routine 2. Inputs in Algorithm 3 are $n = 9$; $M_i = 1000$ for $i = 1, \ldots, n$; $T = 3000$; $t = 600$; $\alpha = 0.2$.

Strategy				Solution	Time (s)
Branching		Fixing			
name	arg	name	arg		
`Naive`	ϕ	`Naive`	ϕ	Intractable	3000
		`SelectAll`	ϕ	[3, 7, 919, 2179, 4159, 13399, 37039, 301759, 1291159]	**113**
		`ExcludeOne`	[1]	Intractable	3000
			[2]	Intractable	3000
			[3]	Intractable	3000
			[4]	Intractable	3000
			[5]	Intractable	3000
			[6]	Intractable	3000
			[7]	Intractable	3000
			[8]	Intractable	3000
		`ExcludeTwo`	[1, 2]	Intractable	3000
			[1, 3]	Intractable	3000
			[1, 4]	Intractable	3000
			[1, 5]	Intractable	3000
			[1, 6]	Intractable	3000
			[1, 7]	Intractable	3000
			[1, 8]	Intractable	3000
			[2, 3]	Intractable	3000
			[2, 4]	Intractable	3000
			[2, 5]	Intractable	3000
			[2, 6]	Intractable	3000
			[2, 7]	Intractable	3000
			[2, 8]	Intractable	3000
			[3, 4]	Intractable	3000
			[3, 5]	Intractable	3000
			[3, 6]	Intractable	3000
			[3, 7]	Intractable	3000
			[3, 8]	Intractable	3000
			[4, 5]	Intractable	3000
			[4, 6]	Intractable	3000
			[4, 7]	Intractable	3000
			[4, 8]	Intractable	3000
			[5, 6]	Intractable	3000
			[5, 7]	Intractable	3000
			[5, 8]	Intractable	3000
			[6, 7]	Intractable	3000
			[6, 8]	Intractable	3000
			[7, 8]	Intractable	3000
`Modulo`	1	`Naive`	ϕ	Intractable	3000
		`SelectAll`	ϕ	[13, 853, 1129, 1213, 2113, 2389, 7333, 20593, 41113]	**2**

Table S5 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
		`ExcludeOne`	[1]	Intractable	3000
			[2]	Intractable	3000
			[3]	Intractable	3000
			[4]	[13, 853, 1129, 7993, 10009, 20173, 30169, 59509, 106453]	123
			[5]	Intractable	3000
			[6]	[13, 853, 1129, 1213, 2113, 20593, 41113, 72733, 335173]	1852
			[7]	[13, 853, 1129, 1213, 2113, 2389, 20593, 41113, 72733]	67
			[8]	[13, 853, 1129, 1213, 2113, 2389, 7333, 20593, 41113]	37
		`ExcludeTwo`	[1, 2]	Intractable	3000
			[1, 3]	Intractable	3000
			[1, 4]	Intractable	3000
			[1, 5]	Intractable	3000
			[1, 6]	Intractable	3000
			[1, 7]	Intractable	3000
			[1, 8]	Intractable	3000
			[2, 3]	Intractable	3000
			[2, 4]	Intractable	3000
			[2, 5]	[13, 7333, 16573, 19009, 24469, 26053, 28429, 75109, 113329]	682
			[2, 6]	Intractable	3000
			[2, 7]	Intractable	3000
			[2, 8]	Intractable	3000
			[3, 4]	Intractable	3000
			[3, 5]	Intractable	3000
			[3, 6]	[13, 853, 3313, 11701, 36973, 53281, 60601, 135601, 159721]	1619
			[3, 7]	Intractable	3000
			[3, 8]	Intractable	3000
			[4, 5]	Intractable	3000
			[4, 6]	Intractable	3000
			[4, 7]	Intractable	3000
			[4, 8]	[13, 853, 1129, 7993, 10009, 20173, 30169, 59509, 106453]	1518
			[5, 6]	Intractable	3000
			[5, 7]	Intractable	3000
			[5, 8]	Intractable	3000
			[6, 7]	Intractable	3000
			[6, 8]	Intractable	3000
			[7, 8]	[13, 853, 1129, 1213, 2113, 2389, 7333, 20593, 41113]	483
`Modulo`	5	`Naive`	ϕ	Intractable	3000
		`SelectAll`	ϕ	[5, 17, 881, 1181, 1637, 14957, 24197, 131297, 184721]	**9**
		`ExcludeOne`	[1]	[2909, 3917, 6389, 44357, 49937, 63809, 65609, 134369, 414077]	2595
			[2]	Intractable	3000
			[3]	Intractable	3000
			[4]	Intractable	3000

Table S5 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[5]	Intractable	3000
			[6]	Intractable	3000
			[7]	Intractable	3000
			[8]	[5, 17, 881, 1181, 1637, 14957, 24197, 131297, 184721]	731
		`ExcludeTwo`	[1, 2]	Intractable	3000
			[1, 3]	Intractable	3000
			[1, 4]	Intractable	3000
			[1, 5]	Intractable	3000
			[1, 6]	Intractable	3000
			[1, 7]	Intractable	3000
			[1, 8]	Intractable	3000
			[2, 3]	Intractable	3000
			[2, 4]	Intractable	3000
			[2, 5]	Intractable	3000
			[2, 6]	Intractable	3000
			[2, 7]	Intractable	3000
			[2, 8]	Intractable	3000
			[3, 4]	Intractable	3000
			[3, 5]	Intractable	3000
			[3, 6]	Intractable	3000
			[3, 7]	Intractable	3000
			[3, 8]	Intractable	3000
			[4, 5]	Intractable	3000
			[4, 6]	Intractable	3000
			[4, 7]	Intractable	3000
			[4, 8]	Intractable	3000
			[5, 6]	Intractable	3000
			[5, 7]	Intractable	3000
			[5, 8]	Intractable	3000
			[6, 7]	Intractable	3000
			[6, 8]	[5, 17, 881, 1181, 1637, 44537, 51341, 92357, 107621]	505
			[7, 8]	[5, 17, 881, 1181, 1637, 14957, 24197, 131297, 184721]	3089
`Modulo`	7	`Naive`	ϕ	Intractable	3000
		`SelectAll`	ϕ	[7, 19, 907, 1699, 1747, 3967, 4759, 16519, 23167]	**2**
		`ExcludeOne`	[1]	[4987, 13567, 28771, 45307, 75211, 111427, 397687, 405967, 677587]	2088
			[2]	Intractable	3000
			[3]	Intractable	3000
			[4]	Intractable	3000
			[5]	Intractable	3000
			[6]	[7, 19, 907, 1699, 1747, 15727, 23167, 32359, 41479]	27
			[7]	[7, 19, 907, 1699, 1747, 3967, 16519, 23167, 101527]	186
			[8]	[7, 19, 907, 1699, 1747, 3967, 4759, 16519, 23167]	11

Table S5 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
		`ExcludeTwo`	[1, 2]	Intractable	3000
			[1, 3]	Intractable	3000
			[1, 4]	Intractable	3000
			[1, 5]	Intractable	3000
			[1, 6]	Intractable	3000
			[1, 7]	Intractable	3000
			[1, 8]	[4987, 13567, 28771, 45307, 75211, 111427, 116191, 280411, 397687]	2181
			[2, 3]	Intractable	3000
			[2, 4]	Intractable	3000
			[2, 5]	Intractable	3000
			[2, 6]	Intractable	3000
			[2, 7]	Intractable	3000
			[2, 8]	Intractable	3000
			[3, 4]	Intractable	3000
			[3, 5]	Intractable	3000
			[3, 6]	Intractable	3000
			[3, 7]	Intractable	3000
			[3, 8]	Intractable	3000
			[4, 5]	Intractable	3000
			[4, 6]	Intractable	3000
			[4, 7]	Intractable	3000
			[4, 8]	Intractable	3000
			[5, 6]	Intractable	3000
			[5, 7]	Intractable	3000
			[5, 8]	[7, 19, 907, 1699, 15427, 16519, 32359, 81619, 101527]	1841
			[6, 7]	[7, 19, 907, 1699, 1747, 16519, 23167, 32359, 101527]	1731
			[6, 8]	[7, 19, 907, 1699, 1747, 15727, 23167, 32359, 41479]	85
			[7, 8]	[7, 19, 907, 1699, 1747, 3967, 4759, 16519, 23167]	121
`Modulo`	11	`Naive`	ϕ	Intractable	3000
		`SelectAll`	ϕ	[11, 23, 911, 1283, 2711, 2843, 3803, 24971, 28631]	**2**
		`ExcludeOne`	[1]	[3083, 4919, 5099, 9323, 21563, 34403, 87443, 97943, 584183]	3087
			[2]	Intractable	3000
			[3]	Intractable	3000
			[4]	Intractable	3000
			[5]	Intractable	3000
			[6]	Intractable	3000
			[7]	[11, 23, 911, 1283, 2711, 2843, 24971, 28631, 406403]	2639
			[8]	[11, 23, 911, 1283, 2711, 2843, 3803, 24971, 28631]	17
		`ExcludeTwo`	[1, 2]	Intractable	3000
			[1, 3]	Intractable	3000
			[1, 4]	[4919, 8219, 24419, 30539, 36599, 42719, 72287, 89759, 129419]	2060
			[1, 5]	Intractable	3000

Table S5 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[1, 6]	Intractable	3000
			[1, 7]	Intractable	3000
			[1, 8]	Intractable	3000
			[2, 3]	Intractable	3000
			[2, 4]	Intractable	3000
			[2, 5]	Intractable	3000
			[2, 6]	Intractable	3000
			[2, 7]	Intractable	3000
			[2, 8]	Intractable	3000
			[3, 4]	Intractable	3000
			[3, 5]	Intractable	3000
			[3, 6]	Intractable	3000
			[3, 7]	Intractable	3000
			[3, 8]	Intractable	3000
			[4, 5]	Intractable	3000
			[4, 6]	Intractable	3000
			[4, 7]	Intractable	3000
			[4, 8]	Intractable	3000
			[5, 6]	Intractable	3000
			[5, 7]	Intractable	3000
			[5, 8]	Intractable	3000
			[6, 7]	[11, 23, 911, 1283, 2711, 24971, 34031, 57383, 117503]	2525
			[6, 8]	[11, 23, 911, 1283, 2711, 24971, 34031, 57383, 117503]	2388
			[7, 8]	[11, 23, 911, 1283, 2711, 2843, 3803, 24971, 28631]	54

Table S6: Computational results for solving model (21) using the fixing strategies in Routine 1 and the branching strategies in Routine 2. Inputs in Algorithm 3 are $n = 10$; $M_i = 1000$ for $i = 1, \ldots, n$; $T = 6000$; $t = 3000$; $\alpha = 0.2$.

Strategy				Solution	Time (s)
Branching		Fixing			
name	arg	name	arg		
`Naive`	ϕ	`Naive`	ϕ	Intractable	6000
		`SelectAll`	ϕ	[3, 7, 919, 2179, 4159, 13399, 37039, 301759, 1291159, 1765759]	**156**
		`ExcludeOne`	[1]	Intractable	6000
			[2]	Intractable	6000
			[3]	Intractable	6000
			[4]	Intractable	6000
			[5]	Intractable	6000
			[6]	Intractable	6000
			[7]	Intractable	6000
			[8]	Intractable	6000
			[9]	Intractable	6000
		`ExcludeTwo`	[1, 2]	Intractable	6000
			[1, 3]	Intractable	6000
			[1, 4]	Intractable	6000
			[1, 5]	Intractable	6000
			[1, 6]	Intractable	6000
			[1, 7]	Intractable	6000
			[1, 8]	Intractable	6000
			[1, 9]	Intractable	6000
			[2, 3]	Intractable	6000
			[2, 4]	Intractable	6000
			[2, 5]	Intractable	6000
			[2, 6]	Intractable	6000
			[2, 7]	Intractable	6000
			[2, 8]	Intractable	6000
			[2, 9]	Intractable	6000
			[3, 4]	Intractable	6000
			[3, 5]	Intractable	6000
			[3, 6]	Intractable	6000
			[3, 7]	Intractable	6000
			[3, 8]	Intractable	6000
			[3, 9]	Intractable	6000
			[4, 5]	Intractable	6000
			[4, 6]	Intractable	6000
			[4, 7]	Intractable	6000
			[4, 8]	Intractable	6000
			[4, 9]	Intractable	6000
			[5, 6]	Intractable	6000
			[5, 7]	Intractable	6000
			[5, 8]	Intractable	6000

Table S6 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[5, 9]	Intractable	6000
			[6, 7]	Intractable	6000
			[6, 8]	Intractable	6000
			[6, 9]	Intractable	6000
			[7, 8]	Intractable	6000
			[7, 9]	Intractable	6000
			[8, 9]	Intractable	6000
`Modulo`	1	`Naive`	ϕ	Intractable	6000
		`SelectAll`	ϕ	[13, 853, 1129, 1213, 2113, 2389, 7333, 20593, 41113, 335173]	**18**
		`ExcludeOne`	[1]	Intractable	6000
			[2]	Intractable	6000
			[3]	Intractable	6000
			[4]	Intractable	6000
			[5]	Intractable	6000
			[6]	Intractable	6000
			[7]	[13, 853, 1129, 1213, 2113, 2389, 41113, 72733, 85669, 335173]	1147
			[8]	Intractable	6000
			[9]	[13, 853, 1129, 1213, 2113, 2389, 7333, 20593, 41113, 335173]	2394
		`ExcludeTwo`	[1, 2]	Intractable	6000
			[1, 3]	Intractable	6000
			[1, 4]	Intractable	6000
			[1, 5]	Intractable	6000
			[1, 6]	Intractable	6000
			[1, 7]	Intractable	6000
			[1, 8]	Intractable	6000
			[1, 9]	Intractable	6000
			[2, 3]	Intractable	6000
			[2, 4]	Intractable	6000
			[2, 5]	Intractable	6000
			[2, 6]	Intractable	6000
			[2, 7]	Intractable	6000
			[2, 8]	Intractable	6000
			[2, 9]	Intractable	6000
			[3, 4]	Intractable	6000
			[3, 5]	Intractable	6000
			[3, 6]	Intractable	6000
			[3, 7]	Intractable	6000
			[3, 8]	Intractable	6000
			[3, 9]	Intractable	6000
			[4, 5]	Intractable	6000
			[4, 6]	Intractable	6000
			[4, 7]	Intractable	6000

Table S6 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[4, 8]	Intractable	6000
			[4, 9]	Intractable	6000
			[5, 6]	Intractable	6000
			[5, 7]	Intractable	6000
			[5, 8]	Intractable	6000
			[5, 9]	Intractable	6000
			[6, 7]	Intractable	6000
			[6, 8]	Intractable	6000
			[6, 9]	Intractable	6000
			[7, 8]	Intractable	6000
			[7, 9]	Intractable	6000
			[8, 9]	Intractable	6000
`Modulo`	5	`Naive`	ϕ	Intractable	6000
		`SelectAll`	ϕ	[5, 17, 881, 1181, 1637, 14957, 24197, 131297, 184721, 6353021]	**299**
		`ExcludeOne`	[1]	Intractable	6000
			[2]	Intractable	6000
			[3]	Intractable	6000
			[4]	Intractable	6000
			[5]	Intractable	6000
			[6]	Intractable	6000
			[7]	Intractable	6000
			[8]	Intractable	6000
			[9]	Intractable	6000
		`ExcludeTwo`	[1, 2]	Intractable	6000
			[1, 3]	Intractable	6000
			[1, 4]	Intractable	6000
			[1, 5]	Intractable	6000
			[1, 6]	Intractable	6000
			[1, 7]	Intractable	6000
			[1, 8]	Intractable	6000
			[1, 9]	Intractable	6000
			[2, 3]	Intractable	6000
			[2, 4]	Intractable	6000
			[2, 5]	Intractable	6000
			[2, 6]	Intractable	6000
			[2, 7]	Intractable	6000
			[2, 8]	Intractable	6000
			[2, 9]	Intractable	6000
			[3, 4]	Intractable	6000
			[3, 5]	Intractable	6000
			[3, 6]	Intractable	6000
			[3, 7]	Intractable	6000

Table S6 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[3, 8]	Intractable	6000
			[3, 9]	Intractable	6000
			[4, 5]	Intractable	6000
			[4, 6]	Intractable	6000
			[4, 7]	Intractable	6000
			[4, 8]	Intractable	6000
			[4, 9]	Intractable	6000
			[5, 6]	Intractable	6000
			[5, 7]	Intractable	6000
			[5, 8]	Intractable	6000
			[5, 9]	Intractable	6000
			[6, 7]	Intractable	6000
			[6, 8]	Intractable	6000
			[6, 9]	Intractable	6000
			[7, 8]	Intractable	6000
			[7, 9]	Intractable	6000
			[8, 9]	Intractable	6000
`Modulo`	7	`Naive`	ϕ	Intractable	6000
		`SelectAll`	ϕ	[7, 19, 907, 1699, 1747, 3967, 4759, 16519, 23167, 4891507]	**221**
		`ExcludeOne`	[1]	Intractable	6000
			[2]	Intractable	6000
			[3]	Intractable	6000
			[4]	Intractable	6000
			[5]	Intractable	6000
			[6]	Intractable	6000
			[7]	Intractable	6000
			[8]	Intractable	6000
			[9]	Intractable	6000
		`ExcludeTwo`	[1, 2]	Intractable	6000
			[1, 3]	Intractable	6000
			[1, 4]	Intractable	6000
			[1, 5]	Intractable	6000
			[1, 6]	Intractable	6000
			[1, 7]	Intractable	6000
			[1, 8]	Intractable	6000
			[1, 9]	Intractable	6000
			[2, 3]	Intractable	6000
			[2, 4]	Intractable	6000
			[2, 5]	Intractable	6000
			[2, 6]	Intractable	6000
			[2, 7]	Intractable	6000
			[2, 8]	Intractable	6000

Table S6 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[2, 9]	Intractable	6000
			[3, 4]	Intractable	6000
			[3, 5]	Intractable	6000
			[3, 6]	Intractable	6000
			[3, 7]	Intractable	6000
			[3, 8]	Intractable	6000
			[3, 9]	Intractable	6000
			[4, 5]	Intractable	6000
			[4, 6]	Intractable	6000
			[4, 7]	Intractable	6000
			[4, 8]	Intractable	6000
			[4, 9]	Intractable	6000
			[5, 6]	Intractable	6000
			[5, 7]	Intractable	6000
			[5, 8]	Intractable	6000
			[5, 9]	Intractable	6000
			[6, 7]	Intractable	6000
			[6, 8]	Intractable	6000
			[6, 9]	Intractable	6000
			[7, 8]	Intractable	6000
			[7, 9]	Intractable	6000
			[8, 9]	Intractable	6000
`Modulo`	11	`Naive`	ϕ	Intractable	6000
		`SelectAll`	ϕ	[11, 23, 911, 1283, 2711, 2843, 3803, 24971, 28631, 406403]	**22**
		`ExcludeOne`	[1]	Intractable	6000
			[2]	Intractable	6000
			[3]	Intractable	6000
			[4]	Intractable	6000
			[5]	Intractable	6000
			[6]	Intractable	6000
			[7]	Intractable	6000
			[8]	Intractable	6000
			[9]	[11, 23, 911, 1283, 2711, 2843, 3803, 24971, 28631, 406403]	2664
		`ExcludeTwo`	[1, 2]	Intractable	6000
			[1, 3]	Intractable	6000
			[1, 4]	Intractable	6000
			[1, 5]	Intractable	6000
			[1, 6]	Intractable	6000
			[1, 7]	Intractable	6000
			[1, 8]	Intractable	6000
			[1, 9]	Intractable	6000
			[2, 3]	Intractable	6000

Table S6 – *Continued from previous page*

Strategy				Solution	Time (s)
Branching		**Fixing**			
name	**arg**	**name**	**arg**		
			[2, 4]	Intractable	6000
			[2, 5]	Intractable	6000
			[2, 6]	Intractable	6000
			[2, 7]	Intractable	6000
			[2, 8]	Intractable	6000
			[2, 9]	Intractable	6000
			[3, 4]	Intractable	6000
			[3, 5]	Intractable	6000
			[3, 6]	Intractable	6000
			[3, 7]	Intractable	6000
			[3, 8]	Intractable	6000
			[3, 9]	Intractable	6000
			[4, 5]	Intractable	6000
			[4, 6]	Intractable	6000
			[4, 7]	Intractable	6000
			[4, 8]	Intractable	6000
			[4, 9]	Intractable	6000
			[5, 6]	Intractable	6000
			[5, 7]	Intractable	6000
			[5, 8]	Intractable	6000
			[5, 9]	Intractable	6000
			[6, 7]	Intractable	6000
			[6, 8]	Intractable	6000
			[6, 9]	Intractable	6000
			[7, 8]	Intractable	6000
			[7, 9]	Intractable	6000
			[8, 9]	Intractable	6000