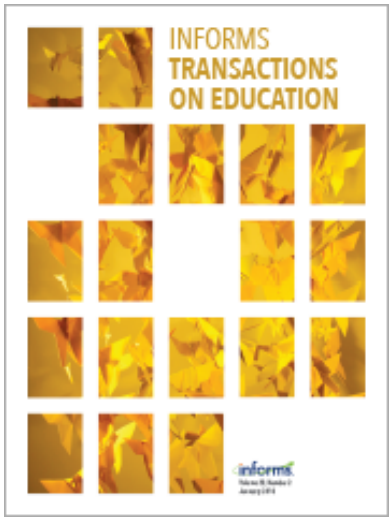

INFORMS Transactions on Education

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Puzzle—Solving the n-Fractions Puzzle as a Constraint Programming Problem

Arnaud Malapert, Julien Provillard

To cite this article:
Arnaud Malapert, Julien Provillard (2018) Puzzle—Solving the n-Fractions Puzzle as a Constraint Programming Problem. INFORMS Transactions on Education ():. https://doi.org/10.1287/ited.2017.0193

http://pubsonline.informs.org/journal/ited/

INFORMS TRANSACTIONS ON EDUCATION
Vol. 19, No. 1, September 2018, pp. 48–55
ISSN 1532-0545 (online)

Puzzle

Solving the n-Fractions Puzzle as a Constraint Programming Problem

Arnaud Malapert,[a] **Julien Provillard**[a]

[a] Université Côte d'Azur, CNRS, I3S, France

Contact: arnaud.malapert@unice.fr, http://orcid.org/0000-0003-0099-479X (AM); julien.provillard@unice.fr (JP)

Received: June 16, 2017
Revised: November 6, 2017; December 4, 2017
Accepted: December 12, 2017
Published Online in Articles in Advance: May 2, 2018

https://doi.org/10.1287/ited.2017.0193

Abstract. The aim in solving puzzles is to find the solution using several clues and restrictions. In this paper, we solve a numerical puzzle, the n-fractions puzzle, by constraint programming. The *n-fractions puzzle* is problem 41 of the CSPLib, a library of test problems for constraint solvers. Models referenced in the CSPLib return invalid solutions as soon as the number n of fractions exceeds five. To solve the n-fractions puzzle, we first provide an upper bound for the unsatisfiability inspired by constraint filtering techniques. Then we propose two new constraint programming models that exploit the integer factorization of the fractions' denominators and their lowest common multiple. The proposed models can solve up to the 19-fractions puzzle within a few minutes and without returning invalid solutions. Some restrictions of the models that eliminate invalid solutions still allow them to solve larger n-fractions puzzles, even if the solving times increase. At the end, only six n-fractions puzzles remain open.

Keywords: puzzles • constraint programming • teaching modeling • teaching optimization

1. Introduction

Constraint programming (CP) has attracted much attention among experts from many areas because of its potential for solving hard real-life problems. For an extensive review on CP, see the handbook by Rossi et al. (2006). A *constraint satisfaction problem* (CSP) consists of a set of variables defined by a corresponding set of possible values (the domains) and a set of constraints. A *constraint* is a relation between a subset of variables that restricts the possible values that variables can simultaneously take. The important feature of constraints is their declarative manner, i.e., they only specify which relationship must hold.

Solutions can be found by systematically searching through the possible assignments of values to variables. A *backtracking scheme* incrementally extends a partial assignment that specifies consistent values for some of the variables, toward a complete solution, by repeatedly choosing a value for another variable. If a partial assignment violates any of the constraints, backtracking is performed to the most recent partial assignment where alternatives remain.

A filtering algorithm is associated with each constraint; it removes inconsistent values from the domains of the variables, i.e., assignments that cannot belong to a solution of the constraint. Constraints are handled through a *constraint propagation mechanism* which reduces the domains of variables until a global fixpoint is reached (i.e., no more domain reductions are possible). In fact, a constraint specifies which relationship must hold and its filtering algorithm is the computational procedure that enforces the relationship.

CSPs are usually solved by a combination of a backtracking scheme and consistency techniques that allow the search space to be completely and efficiently explored. The propagation mechanism facilitates reduction of the variable domains and the pruning of the search space whereas the branching strategy can improve the detection of solutions (or failures for unsatisfiable problems).

CSPLib (1999) is a library of test problems for constraint solvers. The *n-fractions puzzle* (Frisch et al. 1999) is problem 41 of the CSPLib. The original fractions puzzle is specified as follows: find nine distinct non-zero digits that satisfy

$$\frac{A}{BC} + \frac{D}{EF} + \frac{G}{HI} = 1,$$

where BC is shorthand for $10B + C$, EF for $10E + F$, and HI for $10H + I$.

A simple generalization is as follows: find $3n$ nonzero digits satisfying

$$\sum_{i=1}^{n} \frac{x_i}{y_i z_i} = 1, \tag{1}$$

where $y_i z_i$ is shorthand for $10y_i + z_i$ and the number of occurrences of each digit in 1..9 is between 1 and $\lceil n/3 \rceil$. Since each fraction is at least $1/99$, this family of problems has solutions for at most $n \leq 99$. We are interested in finding the greatest n such that at least one n-fractions exists.

In the related work, the *n-fractions puzzle* is considered as a toy problem for illustrating the interest of symmetry breaking rules. Schulte and Smolka (2004) illustrate on the 3-fractions puzzle the advantages of using redundant constraints implied by a symmetry breaking rule (the fractions are sorted in non-decreasing order). Frisch et al. (2001) use proof planning to generate implied algebraic constraints that improve the solution of the previous model. Frisch et al. (2004) study the strength of the implied constraints derivable from symmetry breaking rules (the fractions are sorted in nondecreasing or lexicographic orders). The results on the 3-fractions puzzle show that weaker symmetry breaking can sometimes outperform a stronger scheme when implied constraints are added.

Unfortunately, the two referenced models do not scale because they return invalid solutions as soon as n exceeds five. The *division model* handles the Equation (1) with floating-point arithmetic, and therefore returns invalid solutions because of rounding errors. Furthermore, many CP solvers do not handle floating-point numbers, and even fewer fractional or real numbers.

The *product model* only needs integer arithmetic because the Equation (1) is reformulated as follows:

$$\sum_{i=1}^{n} \left(x_i \prod_{k \neq i} y_k z_k \right) = \prod_{i=1}^{n} y_k z_k \tag{2}$$

The main drawback is that the right member grows exponentially with the number n of fractions. So the computation of the right member leads to integer overflow. The right member overflows a 32-bits integer from $n = 6$.

Although the CSPLib has a research purpose, it is also a source of teaching material for many. We aim to propose better constraint models because the n-fractions puzzle is good teaching material. Indeed, the n-fractions puzzle is already about constraint modeling (symmetry breaking, redundant constraints, . . .). It is also about floating-point and integer arithmetic that causes errors when solving the models. These errors are so frequent that the models quickly become ineffective by returning too many invalid solutions. Since the number representation in computing is an issue, the new models use the prime factorization from number theory. The formal proof of the upper bound is inspired by constraint filtering techniques for summing and counting values. So the proposed approach for the puzzle is intended for students finishing their bachelor degree or passing their master degree, following an introduction to CP. In the meantime, we want to close the n-fractions puzzle by finding a fraction or proving that none exists.

The new *integer factorization models* are based on Equation (3) instead of Equation (1) or (2):

$$\sum_{i=1}^{n} \left(x_i \times \frac{L}{y_i z_i} \right) = L, \tag{3}$$

where L is the lowest common multiple (LCM) of $y_i z_i$ $(i = 1, \ldots, n)$, and therefore $L/(y_i z_i)$ is an integer. In theory, the right member L still grows exponentially, and leads to integer overflow. In practice, we show that the growth is significantly slower, and makes it possible to find large n-fractions. Briefly, these new models exploit the prime factorization of the denominators to compute the lowest common multiple L. In number theory, prime factors of a positive integer are the prime numbers that exactly divide that integer. For a prime factor p of n, the multiplicity of p is the largest exponent m for which p^m divides n exactly. The prime factorization of a positive integer is a list of the integer's prime factors, together with their multiplicities. The process of determining these factors is called integer factorization. The fundamental theorem of arithmetic says that every positive integer has a single unique prime factorization. For example, $300 = 2^2 \times 3 \times 5^2$, in which factors 2, 3, and 5 have multiplicities of 2, 1, and 2, respectively.

The paper is organized as follows. Section 2 gives a formal proof that the n-fractions puzzle is unsatisfiable from $n \geq 45$. Section 3 presents the new CP models based on integer factorization. Section 4 gives experimental results for parameter settings and performance evaluation. Section 5 describes how the puzzle is used in an introductory course to operations research (OR) for M.Sc. students in computer science. At the end, only six n-fractions remain open for $n \in \{36, 39, 41, 42, 43, 44\}$.

2. An Upper Bound for the Unsatisfiability

In this section we prove that the n-fractions puzzle does not admit a solution for $n \geq 45$. For a given n, we exhibit a lower bound on the sum $S = \sum_{i=1}^{n} x_i/(y_i z_i)$ for digits x_i, y_i, and z_i, which respects the counting constraints of the puzzle. As soon this lower bound exceeds 1, the puzzle becomes unsatisfiable.

We call a *valid digits assignment* of the variables x_i, y_i, and z_i an assignment that respects one of the counting constraints of the n-fractions puzzle: each digit appears at most $\lceil n/3 \rceil$ times. We call a *minimal digits assignment*

Table 1. Selected Digits in a Minimal Digits Assignment

n	1	2	3	4	5	6	7	8	9
3	1	1	1	1	1	1	1	1	1
4	2	2	0	0	0	2	2	2	2
5	2	2	1	0	2	2	2	2	2
6	2	2	2	2	2	2	2	2	2
7	3	3	1	0	2	3	3	3	3
8	3	3	2	1	3	3	3	3	3
$3k$	k	k	k	k	k	k	k	k	k
$3k+1$	$k+1$	$k+1$	$k-1$	$k-3$	$k+1$	$k+1$	$k+1$	$k+1$	$k+1$
$3k+2$	$k+1$	$k+1$	k	$k-1$	$k+1$	$k+1$	$k+1$	$k+1$	$k+1$

a valid digit assignment that minimizes the sum $S = \sum_{i=1}^{n} x_i/(y_i z_i)$.

To find a lower bound on S, we look for the mandatory conditions on a minimal digits assignment. We first determine which digits are allowed in a minimal digits assignment. Then, we state several necessary conditions that all together allow the model to compute the lower bound. Each of these constraints is based on the same idea. We consider an assignment that verifies some assumptions and we try to swap two digits. If the resulting sum is less than the initial sum, a minimal digits assignment should not satisfy the same assumptions. For now, x, y, z, x', y', and z' denote six non-zero digits.

In a valid assignment, any digit appears at most $\lceil n/3 \rceil$ times. We must choose $3n$ digits among $9\lceil n/3 \rceil$ allowed digits. If n is a multiple of 3, there is no choice. Yet in the other cases, we must identify the possible digits in a minimal digits assignment.

Observation 1. If $x' < x$ then $x/(yz) > x'/(yz)$.
If $y < y'$ then $x/(yz) > x/(y'z)$.
If $z < z'$ then $x/(yz) > x/(yz')$.

Observation 1 just stresses that to minimize a fraction, one has to minimize its numerator and maximize digits of its denominator. Then if d is an unselected digit in a valid assignment and $d < x_i$ for some i, it is possible to swap the two values to reduce S. Conversely, if d is an unselected digit and $d > y_i$ or $d > z_i$ for some i, a swap also reduces S. Therefore, in a minimal digits assignment the smallest unselected digit is not less than the largest x_i and the largest unselected digit is not greater than the smallest y_i or the smallest z_i. To conclude, in a minimal digits assignment the n smallest allowed values are assigned to the variables x_i while the $2n$ largest allowed values are assigned to the variables y_i and z_i. Table 1 summarizes the selected digits in a minimal digits assignment depending on the three congruences of n modulo 3. Note that all digits are selected at least once in a minimal digits assignment for $n \geq 11$ which leads to Corollary 1.

Corollary 1. *A minimal digits assignment respects all the counting constraints of the n-fractions puzzle for $n \geq 11$: Any digit appears at least once and at most $\lceil n/3 \rceil$ times.*

We now state several observations that induce necessary conditions for a minimal digits assignment. We are particularly interested in local behavior involving one or two fractions.

Observation 2. If $y < z$ then $x/(yz) > x/(zy)$.

This observation is straightforward. To minimize a fraction, we must maximize its denominator. Therefore, the tens digit of the denominator should be not less than the ones digit. The next observation extends the results to a sum of two fractions. Then, the same holds for an arbitrary sum of fractions.

Observation 3. If $z \leq y < z' \leq y'$ then $x/(yz) + x'/(y'z') > x/(z'z) + x'/(y'y)$.

Let A denote $x/(yz) + x'/(y'z') - (x/(z'z) + x'/(y'y))$. The observation is true if and only if $A > 0$ under the assumptions $z \leq y < z' \leq y'$.

By definition,

$$A = \frac{x}{10y+z} + \frac{x'}{10y'+z'} - \frac{x}{10z'+z} - \frac{x'}{10y'+y}.$$

We reduce the fraction to the same denominator and we factorize the numerator:

$$A = \frac{(z'-y)B}{(10y+z)(10y'+z')(10z'+z)(10y'+y)},$$

where $B = 100xyy' + 10xyz' + 1{,}000xy'^2 + 100xy'z' - 10x'yz - 100x'yz' - x'z^2 - 10x'zz'$.

As $z' > y$ by assumption and the denominator of A is positive, A and B have the same sign.

By assumption $y' \geq z'$ and $z \leq y$, we can obtain a lower bound on B by replacing all occurrences of y' by z' (resp. z by y) as y' always appears in non-negative terms (resp. z always appears in non-positive terms):

$$B \geq 100xyz' + 10xyz' + 1{,}000xz'^2 + 100xz'^2 - 10x'y^2 - 100x'yz' - x'y^2 - 10x'yz'.$$

The lower bound can be factorized and then

$$B \geq 11(10z'+y)(10xz' - x'y).$$

As $x \geq 1$ and $x' < 10$, we can replace x by 1 and x' by 10 so that

$$B > 11(10z'+y)(10z'-10y) = 110(10z'+y)(z'-y).$$

Again, by assumption $z' > y$, then $B > 0$, and finally $A > 0$.

Observation 3 enforces that for any pair of fractions in a minimal digits assignment the tens digit of a denominator should not be less than the ones digit of a denominator, i.e., $\max(z_i, z_j) \leq \min(y_i, y_j)$ for each couple (i, j). Applying the result to each possible pair of fractions, we get $\max\{z_i\} \leq \min\{y_i\}$. Among the $3n$

chosen digits, we know that the n lowest go to the numerators. Then, the n greatest go to the tens digits of the denominators. As we know which digits are selected in a minimal digits assignment, we obtain the following corollary:

Corollary 2. *In a minimal digits assignment,* $1 \le x_i \le 3$, $4 \le z_i \le 7$ *and* $7 \le y_i \le 9$. *Moreover, if* $n = 0 \pmod 3$ *then* $z_i \le 6$.

The next observation shows how the digits choice for the numerators in a minimal digits assignment impacts the digits choice for the denominators. This observation is a particular case of the rearrangement inequality theorem (Hardy et al. 1952).

Observation 4. If $x < x'$ and $yz > y'z'$ then $x/(yz) + x'/(y'z') > x/(y'z') + x'/(yz)$.

$$\frac{x}{yz} + \frac{x'}{y'z'} - \left(\frac{x}{y'z'} + \frac{x'}{yz}\right) = (x' - x)\left(\frac{1}{y'z'} - \frac{1}{yz}\right) > 0.$$

Observation 4 indicates that in a minimal digits assignment, the lowest denominators go with the lowest numerators. Note that in this observation we do not swap two digits but whole denominators. We can only deduce that the lowest tens digits of the denominators go with the lowest numerators. The same does not hold for the ones digit of the denominators. For instance, $2/75 + 3/96 > 2/76 + 3/95$.

We can now state the main result of the section.

Proposition 1. *If* $n \ge 45$, *then* $S > 1$.

We assume that $n \ge 11$ so that Corollary 1 applies. This is not a restriction because the n-fractions puzzle is shown satisfiable for $n < 11$ in Section 4. There are three cases in the proof corresponding to the three congruences of n modulo 3. Still the same ideas hold for all. First, the n lowest values (Table 1) are assigned to the x_i in non-decreasing order. Second, the n greatest values are assigned to the y_i in non-decreasing order. Finally, the z_i's are replaced by 6 if $n = 0 \pmod 3$ or 7 otherwise (Corollary 2). Observation 4 ensures these operations give a lower bound on S. Note that it is not possible to obtain a better bound on the ones digits of the numerator as explained in the remarks after Observation 4. Table 2 summarizes the computed lower bound $B(n)$ in function of the congruence of n modulo 3. The last column gives the threshold such that $B(n) > 1$. We can conclude that if $n \ge 45$, $S \ge B(n) > 1$.

Corollary 3. *The n-fractions puzzle has no solution for* $n \ge 45$.

3. Integer Factorization Models

In this section, we present new CP models based on the integer factorization of the denominators of the irreducible fractions. The models state table constraints, counting constraints, sum constraints, symmetry breaking constraints, and redundant constraints.

Table 2. Lower Bound $B(n)$ on S and Threshold on n Such That $B(n) > 1$

	Lower bound $B(n)$ on S	Threshold on n
$n = 3k$	$\left(\frac{1}{76} + \frac{2}{86} + \frac{3}{96}\right)k$	45
$n = 3k+1$	$\left(\frac{1}{77} + \frac{2}{87} + \frac{3}{97}\right)(k-1) + 2\left(\frac{1}{87} + \frac{2}{97}\right)$	46
$n = 3k+2$	$\left(\frac{1}{77} + \frac{2}{87} + \frac{3}{97}\right)k + \frac{1}{87} + \frac{2}{97}$	47

3.1. Fraction Table Constraints

For each fraction i $(i = 1, \dots, n)$, a table constraint defines the allowed assignments (see `in_relation` in Beldiceanu et al. 2017). If some constraints cannot be easily expressed, by a formula or by a regular pattern, a table constraint allows us to specify in extension the combinations of allowed values. Here, the table constraint defines the allowed fractions, their irreducible forms, some fixed precision bounds, and the integer factorization of the irreducible denominator.

The variables x_i, y_i, and z_i define the fraction $x_i/(y_i z_i)$. The variables n_i and d_i represent the numerator and the denominator of the irreducible fraction n_i/d_i (the numerator and denominator are coprime) equal to the original fraction $x_i/(y_i z_i)$. The variables l_i^P and u_i^P, respectively, represent a lower bound and an upper bound on the fraction $x_i/(y_i z_i)$ with a fixed precision P.

$$l_i^P = \left\lfloor P \times \frac{x_i}{y_i z_i} \right\rfloor \qquad u_i^P = \left\lceil P \times \frac{x_i}{y_i z_i} \right\rceil$$

We consider two different models of the prime factorization of the denominator d_i. Let $\mathcal{P}$ denote the prime numbers in the interval $[1, 99]$. Every positive integer $d_i \in [1, 99]$ can be represented in exactly one way as a product of prime powers: $d_i = \prod_{p \in \mathcal{P}} p^{m_i^p}$. The maximum multiplicity of each factor is bounded so that d_i remains lower than 100. In the first model, the scope of the table constraint is the following:

$$\langle x_i, y_i, z_i, n_i, d_i, l_i^P, u_i^P, m_i^2, m_i^3, \dots, m_i^p, \dots, m_i^{89}, m_i^{97} \rangle. \quad \text{(Mult)}$$

Next, since d_i has at most one prime factor greater than 10 (otherwise $d_i > 99$), a compact representation of d_i is given by $d_i = 2^{m_i^2} \times 3^{m_i^3} \times 5^{m_i^5} \times 7^{m_i^7} \times p_i$ where p_i is 1 or a prime in $\mathcal{P}$ greater than 10. In the second model, the scope of the table constraint is the following:

$$\langle x_i, y_i, z_i, n_i, d_i, l_i^P, u_i^P, m_i^2, m_i^3, m_i^5, m_i^7, p_i \rangle. \quad \text{(Fact)}$$

For instance, if the precision P is equal to 100, then the fraction 6/54 is associated to the tuple $\langle x_i = 6, y_i = 5, z_i = 4, n_i = 1, d_i = 9, l_i^P = 11, u_i^P = 12, m_i^2 = 0, m_i^3 = 2, \dots, m_i^p = 0, \dots, m_i^{89} = 0, m_i^{97} = 0 \rangle$ with the factorization Mult, whereas it is associated to the tuple $\langle x_i =$

$6, y_i = 5, z_i = 4, n_i = 1, d_i = 9, l_i^p = 11, u_i^p = 12, m_i^2 = 0, m_i^3 = 2, m_i^5 = 0, m_i^7 = 0, p_i = 1\rangle$ with the factorization Fact.

3.2. Counting Constraints

The number of occurrences of each digit is constrained by stating a conjunction of `among_low_up` constraints (see `among_low_up` in Beldiceanu et al. 2017) which enforces that between 1 and $\lceil n/3 \rceil$ variables of the $\langle x_i, y_i, z_i \rangle_{1 \le i \le n}$ collection are assigned a given digit $v \in [1, 9]$.

$$\texttt{among\_low\_up}\left(1, \left\lceil n/3 \right\rceil, \langle x_1, y_1, z_1, \ldots, x_n, y_n, z_n \rangle, v\right) \quad \forall v \in [1, 9].$$

For instance, $\texttt{among\_low\_up}(1, 2, \langle 9, 2, 4, 5, 2\rangle, 2)$ constraint holds since two values of the collection of values $\langle 9, 2, 4, 5, 2\rangle$ equal two. On the contrary, $\texttt{among\_low\_up}(1, 2, \langle 9, 2, 4, 5, 2\rangle, 3)$ constraint does not hold since no value of the collection of values $\langle 9, 2, 4, 5, 2\rangle$ is equal to three.

3.3. LCM Constraints

Let L be the lowest common multiple (LCM) of the d_i $(i = 1, \ldots, n)$; the Equation (3) is stated with the irreducible fractions:

$$\sum_{i=1}^{n} \left(n_i \times \frac{L}{d_i} \right) = L, \tag{4}$$

where L/d_i is an integer. The lowest common multiple L is also factorized:

$$L = \prod_{p \in \mathcal{P}} p^{m^p}$$

The multiplicities of L are stated as follows:

$$m^p = \max_{1 \le i \le n} m_i^p, \quad \forall p \in \mathcal{P}.$$

Some constraints change when using the factorization Fact:

$$m^p = \max_{1 \le i \le n} \delta_p(p_i) \quad \forall p \in \mathcal{P}, p > 10,$$

where $\delta_p(x)$ is equal to 1 if $x = p$, otherwise 0.

The corollary of the following proposition explains why the integer factorization models can scale better than the product model.

Proposition 2. *If a prime factor p has a multiplicity m in the denominator of a fraction n_i/d_i, then the maximum multiplicity of p in the other fractions is greater than or equal to m.*

The Equation (1) can be rewritten as $n_i/d_i + n/d = 1$ where both fractions are irreducible and $n/d = \sum_{i \ne j} n_j/d_j$. Once again, this can be rewritten as $dn_i + d_i n = dd_i$, and then $dn_i = d_i(d - n)$. So, d_i divides d because n_i and d_i are coprime. Finally, p^m divides d_i which divides d. □

Corollary 4. *If $n > 1$, $\sqrt{\prod_{i=1}^{n} y_k z_k} \ge L$.*

Note that an overflow can happen during domain filtering even if the constraint (3) is consistent.

3.4. Symmetry Breaking Rules

Frisch et al. (2004) study 4 alternatives of symmetry-breaking constraints. First, the fractions $x_i/(y_i z_i)$ are symmetrical. To break this symmetry, ordering constraints IncF may be added. Then, consider arranging the variables in a $3 \times n$ matrix. Given the problem constraints, this matrix has column symmetry: Any pair of columns can be exchanged in a solution to generate a solution. It does not have row symmetry, so all symmetry can be broken by constraining the columns to be lexicographically ordered (Flener et al. 2002). This can be done in six ways, depending on the order of significance of the variables in each column. We consider three alternatives in addition to the ordering constraints:

$$\frac{x_i}{y_i z_i} \le \frac{x_{i+1}}{y_{i+1} z_{i+1}} \quad 1 \le i \le n-1 \tag{IncF}$$

$$(x_i, y_i, z_i) \le_{\text{lex}} (x_{i+1}, y_{i+1}, z_{i+1}) \quad 1 \le i \le n-1 \tag{LexA}$$

$$(y_i, z_i, x_i) \le_{\text{lex}} (y_{i+1}, z_{i+1}, x_{i+1}) \quad 1 \le i \le n-1 \tag{LexB}$$

$$(z_i, x_i, y_i) \le_{\text{lex}} (z_{i+1}, x_{i+1}, y_{i+1}) \quad 1 \le i \le n-1. \tag{LexC}$$

3.5. Redundant Constraints

A generalization of Equation (13) in Frisch et al. (2004) states that the numerators sum is equal to or greater than the minimum denominator. Symmetrically, this sum is equal to or lower than the maximum denominator.

$$\min_{1 \le i \le n} y_i z_i \le \sum_{i=1}^{n} x_i \le \max_{1 \le i \le n} y_i z_i$$

$$\min_{1 \le i \le n} d_i \le \sum_{i=1}^{n} n_i \le \max_{1 \le i \le n} d_i.$$

We also state redundant constraints on the multiplicities of prime factors. Using Proposition 2, a prime factor of an irreducible denominator is common to at least one other. Therefore, L has at most $n/2$ prime factors greater than 10 because each denominator has at most one prime factor greater than 10.

$$\sum_{i=1}^{n} m_i^p \ge 2 \times m^p \quad \forall p \in \mathcal{P}$$

$$\sum_{p \in \mathcal{P}, p > 10} m^p \le \frac{n}{2}.$$

The sum of the n-fractions is bounded by the fixed precision P as follows:

$$\sum_{i=1}^{n} l_i^p \le P \le \sum_{i=1}^{n} u_i^p.$$

A high value of P tightens the lower and upper bounds, and therefore strengthens the constraint propagation, but can slow down the search.

Last, the rule IncF implies redundant ordering constraints over other variables.

$$\frac{n_i}{d_i} \leq \frac{n_{i+1}}{d_{i+1}} \quad l_i^P \leq l_{i+1}^P \quad u_i^P \leq u_{i+1}^P \quad 1 \leq i \leq n-1.$$

4. Experimental Results

This section presents computational experiments conducted to evaluate our approach. All the experiments were conducted on a cluster composed of 72 nodes (1,152 cores) running on CentOS 6.3, each node with 64 GB of RAM and 2 Intel E5-2670 2.60 GHz processors (8 cores). The implementation is based on CP Optimizer bundled in IBM ILOG CPLEX Optimization Studio 12.5 (2012).

First, the parameters of the model are tuned on the 3-fractions puzzle. Second, different search algorithms are compared up to the 19-fractions puzzle. Finally, the best parameter settings are used to find as much n-fractions as possible.

Every solution is checked using exact arithmetic so that only valid solutions are reported. In practice, invalid solutions are only found from the 20-fractions puzzle. For $n \geq 20$, an underapproximation of the search space is explored by posting additional constraints that eliminate all invalid solutions.

4.1. Model Parameter Tuning with the 3-Fractions Puzzle

Following a protocol inspired by Frisch et al. (2004), we analyze three parameters of the model for the 3-fractions, i.e., the precision $P \in \{1\text{E}3, 1\text{E}4\}$, the factorization (Mult and Fact), and the symmetry breaking rules (None, IncF, LexA, LexB, and LexC). For a thorough comparison, the models are compared using 1,000 variable orderings of x_i, y_i, z_i $(i = 1 \ldots 3)$ uniformly drawn among the 9! ones. To remove the effects of finding a good value ordering by chance, the entire search space is searched for each ordering using a sequential depth-first search (DFS). Note that there is a unique 3-fractions with respect to the symmetry.

The performance profile of the two factorization models Mult and Fact is similar, but Fact is faster whereas Mult scales better (see Section 4.2). Here, we only report the results given by models using Mult. Table 3 gives the mean and standard deviation of the number of branches depending on the precision and the symmetry breaking rule. First, the symmetry breaking rules are efficient because they reduce the average number of branches by a factor 3 and their standard deviation by a factor 2. Second, increasing the precision always slightly reduces mean and standard deviation. Third, the performance appears to improve those of Frisch et al. (2004) using a similar experimental protocol. Last, the performance of the symmetry breaking rules is very similar whereas LexB and IncF were the best choices in Frisch et al. (2004).

Table 3. Comparison Among the Different Models of the Number of Branches (Mean and Standard Deviation) Taken to Explore the Entire Search Space of the 3-Fractions Puzzle

Precision	Branches	None	IncF	LexA	LexB	LexC
$P = 1{,}000$	Mean	1,537	584	578	572	548
	Std	868	452	273	590	377
$P = 10{,}000$	Mean	1,498	575	569	563	538
	Std	862	448	273	588	369

Note. Times scale with branches.

4.2. Search Comparison Up to the 19-Fractions Puzzle

Now, searches are compared for finding the first n-fractions for n ranging from 6 to 19. Of course, it becomes impossible to explore the entire search space as n increases. Preliminary experiments showed that a parallel search is more efficient than a sequential search. So, we consider two parallel searches (IBM 2012), i.e., a DFS; and an automatic search (AUTO). Each worker of the parallel DFS executes its own DFS. The workers will not do the same exploration due to some randomness introduced to break ties in decision making. AUTO spreads searches (DFS, DFS with restart, MultiPoint) over the workers. The decision of choosing the search type for a worker is made automatically. Each parallel search uses eight workers with a time limit of two hours. The time limit is the total runtime of all workers.

Next we consider the following set of model parameters: $P = 1\text{E}4$; factorization Mult or Fact; rule LexB or IncF. Table 4 gives the wall-clock time for finding the first n-fractions depending on these parameters where dashes stand for timeouts. The factorization Mult scales better with n, but is slower than Fact The best search is AUTO when using the factorization Mult, but DFS when using Fact. The best choice of symmetry breaking is now IncF, maybe thanks to its redundant ordering constraints. To conclude, these experiments show that the combination of the model and the search algorithm is very important, and that the parameter tuning on small instances is not enough.

4.3. Finding Large n-Fractions

Preliminary experiments showed that the integer factorization models sometimes return invalid n-fractions if n is greater than 20. Recall that an overflow can occur during the constraint propagation even if the constraints are consistent. To avoid any overflow, we state additional constraints that restrict the search space so

Table 4. Comparison of Wall-Clock Times (in Seconds) of Parallel Searches for Finding the First n-Fractions

	Mult				Fact			
	LexB		IncF		LexB		IncF	
n	AUTO	DFS	AUTO	DFS	AUTO	DFS	AUTO	DFS
6	**59.4**	64.5	66.7	74.0	16.5	**16.2**	18.3	17.6
7	**63.4**	71.3	66.2	87.1	17.4	17.4	**16.1**	16.3
8	92.5	93.3	**72.1**	99.8	17.5	**9.3**	38.3	17.0
9	**90.5**	192.0	118.4	166.1	121.0	**9.0**	4,906.2	12.2
10	93.6	182.0	**91.1**	184.1	96.9	**11.6**	—	16.7
11	**120.8**	235.1	127.3	192.0	184.3	16.9	2,733.9	**9.3**
12	321.2	329.7	**185.9**	717.2	—	**49.1**	—	105.9
13	171.7	138.5	**138.3**	150.8	—	**11.2**	—	23.8
14	178.9	1,761.6	**148.6**	258.9	3,933.2	478.9	—	**62.3**
15	555.0	**283.6**	381.0	—	—	—	—	**64.8**
16	203.8	—	**154.5**	628.2	—	1,331.8	—	**131.8**
17	400.1	—	**219.5**	—	—	—	—	**261.7**
18	—	—	**282.0**	—	—	—	—	—
19	1,134.8	—	**387.9**	—	—	—	—	**461.9**

Note. For each model (Mult and Fact), the best wall-clock time is indicated in bold.

that all invalid solutions are excluded. Yet this under-approximation of the search space has a drawback: Some valid solutions can also be excluded. Here, some parameters are fixed based on the previous experiments, i.e., parallel search AUTO, factorization Mult, and rule IncF. The precision P is in $\{1\text{E}4, 1\text{E}5, 1\text{E}6\}$.

First, the maximum value of the LCM variable L involved in the sum constraint (4) is bounded: $L \leq M/O$, where M is the maximum integer allowed in a domain, and O is the overflow reduction factor. In our case, an integer domain is represented via 64-bits double precision floating-point numbers, and O is in $\{1, 1\text{E}3, 1\text{E}6\}$.

Second, the parameter p restricts the prime factors of denominators to those lower or equal to p. It reduces the search space and provides an intensification in a promising part of the search space. Indeed, the largest prime factor of the denominators is 23 among all n-fractions found in this study.

For a given n, the parameters O and P are tried in increasing lexicographic order with a time limit of 12 hours. The process is repeated while no n-fractions has been found. The value of p is manually restricted for the hardest puzzles.

Table 5 gives the wall-clock time and the parameters P, O, and p for which n-fractions were found for n ranging from 20 to 44. Dashes represent an open n-fractions or that the parameter p is not used. All puzzles are solved up to the 35-fractions so that only six puzzles remain open. Clearly, the restrictions allow us to solve large n-fractions puzzles, but wall-clock times grow. The parallelism and randomness of the search algorithm partially explain the weak correlation between wall-clock times and the number n of fractions. Nevertheless, solving the puzzle when n is a multiple of three takes in average 50% time and even fails when n is equal to 36 or 39. A likely cause is that the tighter counting constraints make the puzzle harder.

Table 5. Finding Large n-Fractions in a Restricted Search Space

n	t	P	O	p	n	t	P	O	p
20	435.1	4	0	—	33	23,056.5	6	6	—
21	5,062.3	4	3	—	34	8,434.5	4	6	—
22	679.4	4	0	—	35	26,202.9	6	6	—
23	685.6	4	6	—	36	—	—	—	—
24	23,065.8	4	6	—	37	1,614.3	4	3	—
25	924.3	4	0	—	38	3,880.0	5	3	40
26	2,371.1	5	6	—	39	—	—	—	—
27	13,176.0	6	6	—	40	36,363.7	6	3	—
28	2,705.1	4	3	—	41	—	—	—	—
29	801.2	5	6	—	42	—	—	—	—
30	1,890.8	6	6	—	43	—	—	—	—
31	4,116.5	4	6	37	44	—	—	—	—
32	1,242.9	5	3	31					

Note. P and O are given as $1\text{E}k$.

Last, we posit that the puzzles are unsatisfiable if n is greater or equal to 41. Indeed, the restricted models are often unsatisfiable.

5. Classroom Experience

One of the main reasons for solving this puzzle is that many of our bachelor's and master's students do not fully realize that computer science, and especially operations research or constraint programming, require multidisciplinary skills. Hence our students do

not naturally show a strong interest in mathematics, and rarely put into practice testing methods learned in their software engineering course. This case serves as a real "eye opener" for many students, especially when they realize that their model is mathematically correct, but wrong in practice. When the instructor can position the case in this way, it is relatively easy to teach students about the difficulty of writing good models, the importance of mathematics, and the need for rigorous testing. Our case study has been tested with 35 M.Sc. students, a large fraction of our annual student group, having already earned their B.S. in computer science or, more rarely, in mathematics.

All students were familiar with basic integer arithmetic concepts in mathematics and computer science. The puzzle is given as a group project assignment in an OR and CP introductory course . We use one 90 minute kickstart session in which students visit the CSPLib page, explore available resources (references and models), and implement a first model (usually the division or product model). Then, we give homework to provide a solution checker using exact integer arithmetic (for instance in Python) and to verify their results.

At the beginning of the next session, a debriefing points out that the models return invalid solutions. The top students have already discovered this problem, but have not identified its causes, and do not know how to solve it. So we introduce the core parts of the integer factorization model, especially the fraction table and LCM constraints. We explain why it is more efficient to precompute the integer factorization of a denominator rather than stating constraints to do so. We also warn the students that the model will still be prone to calculation errors, but for larger values of n. Then, the students must implement the integer factorization model and find the largest possible n-fractions. Finally, students submit their model as well as their largest n-fractions.

After submission, we ask the students about the upper bound for the unsatisfiability. They are usually convinced that there is no solution for the largest values of n. So we ask them to propose a relaxation of their model that tightens the upper bound. Keeping counting and fixed precision constraints is a decent and valid relaxation. Indeed, arithmetic errors come from the main equation and the LCM constraints. Finally, we present a simplified version of the proof to illustrate some principles involved in advanced constraint filtering techniques.

6. Conclusion

In spite of its apparent simplicity, the n-fractions puzzle has been shown to be nontrivial because of its combinatorics, and also because of integer arithmetic. Thanks to a theoretical lower bound and to new constraint models based on prime factorization, only six puzzles remain open. The theoretical lower bound proves that the puzzle has no solution for $n \geq 45$ whereas the constraint models efficiently solve the puzzles while avoiding integer arithmetic errors. The classroom experience showed that this puzzle teaches students about the hidden difficulty of "easy" problems, the art of writing good models, the importance of mathematics, and the need for rigorous testing.

Acknowledgments

The authors thank Carine Fedele for her diligent proofreading of this manuscript. They also thank the anonymous referees for their suggestions.

References

Beldiceanu N, Carlsson M, Demassey S (2017) Global constraint catalogue. Accessed April 26, 2018, http://sofdem.github.io/gccat/.

Flener P, Frisch AM, Hnich B, Kiziltan Z, Miguel I, Pearson J, Walsh T (2002) Breaking Row and Column Symmetries in Matrix Models. Van Hentenryck P, ed. *Proc. 8th Internat. Conf. Principles Practice Constraint Programming, CP 2002* (Springer, Berlin), 462–477.

Frisch A, Miguel I, Walsh T (2001) Extensions to proof planning for generating implied constraints. *Proc. Calculemus-01*, 130–141.

Frisch A, Jefferson C, Miguel I, Walsh T (1999) CSPLib problem 041: The n-fractions puzzle. Accessed April 26, 2018, http://www.csplib.org/Problems/prob041.

Frisch AM, Jefferson C, Miguel I (2004) Symmetry breaking as a prelude to implied constraints: A constraint modelling pattern. López de Mántaras R, Saitta L, eds. *Proc. 16th Eur. Conf. Artificial Intelligence, ECAI'2004* (IOS Press, Amsterdam), 171–175.

Hardy GH, Littlewood JE, Pólya G (1952) Inequalities. *Math. Gazette*. 37(321), https://doi.org/10.1017/S0025557200027455.

IBM (2012) IBM ILOG CPLEX optimization studio. Accessed April 26, 2018, http://www.ibm.com/fr-fr/marketplace/ibm-ilog-cplex.

Jefferson C, Miguel I, Hnich B, Walsh T, Gent IP (1999) CSPLib: A problem library for constraints. Accessed April 26, 2018, http://www.csplib.org/.

Rossi F, Van Beek P, Walsh T, eds. (2006) *Handbook of Constraint Programming* (Elsevier, Amsterdam).

Schulte C, Smolka G (2004) Finite domain constraint programming in Oz: A tutorial. Accessed April 26, 2018, http://mozart.github.io/.