Computing the Frequency of Partial Orders

M.R.C. van Dongen (dongen@cs.ucc.ie)

Centre for Efficiency Orientated Languages
Cork Constraint Computation Centre
Computer Science Department
University College Cork

Abstract. In this paper we will study four algorithms for computing the *frequency* of a given partial order. Here the *frequency* of a partial order is the number of standard labellings respecting that partial order. We formulate frequency computation of partial orders as counting the solutions to a CSP (constraint satisfaction problem). The first two algorithms count all solutions to the CSP using backtrack search while maintaining arc consistency. However, large numbers of solutions to CSPs soon make algorithms based on enumeration infeasible. The third and fourth algorithm, to a degree, overcome this problem. They avoid repeatedly solving problems with isomorphic solutions. To be more precise both algorithms avoid labelling a fixed suborder with *different* label sets. In addition the fourth algorithm also avoids repeated labellings of a fixed suborder with the *same* label set. A prototype implementation of the fourth algorithm in ECL^{i}PS^e was significantly more efficient than an enumeration based counting implementation using OPL. Problems that would have taken more than 10^{147} years with OPL require less than a second with our prototype implementation. Finally, we describe how to generalise our algorithms for counting solutions to other classes of problems.

1 Introduction

In this paper we will study four algorithms for computing the *frequency* of partial orders. Here the *frequency* of a partial order $\langle N, \sqsubseteq \rangle$ having finitely many nodes is the number of bijections $l : N \mapsto \mathcal{L}$ such that for all $x, y \in N$ it is true that $x \sqsubseteq y \Longrightarrow l(x) \leq l(y)$, where $\mathcal{L} = \{1, \ldots, |N|\}$ is the *standard label set* of $\langle N, \sqsubseteq \rangle$. The problem of frequency computation is posed in [Schellekens, 2004], where it is called *counting labellings*. The definition of frequency is a generalisation of the definition for the frequency of heap-ordered trees provided in [Sedgewick and Flajolet, 1996, Page 316].

Partial orders have applications to temporal reasoning, planning and scheduling, configuration, process algebras, partially ordered constraint optimisation, and so on. In [Schellekens, 2004] novel techniques are introduced to considerably facilitate the average timing of algorithms. At CEOL (Centre for Efficiency Orientated Languages) our main interest for studying the frequency of partial orders is the application to the average time complexity analysis of algorithms operating on partial orders, where the frequency is used to quantify the denominator in average time expressions.

Our first two algorithms are based on the correspondence between partial orders and constraint satisfaction problems (CSPs). They encode the allowed bijections as the

solutions to an integer valued CSP by imposing inequality relations $\cdot < \cdot$ between all pairs of nodes that are related in the partial order and by imposing an `alldifferent` constraint. Both algorithms enumerate all solutions using search while maintaining arc consistency (MAC). The first algorithm uses a binary encoding of the `alldifferent` constraint. The second algorithm uses a global `alldifferent` constraint. Encoding the problem as a CSP and counting by enumerating solutions improves upon existing special purpose algorithms for frequency computation which were previously developed at CEOL. Unfortunately, a lot of time is spent on computing isomorphic solutions.

As the number of solutions becomes larger it soon becomes impossible to count by enumerating all solutions. For example, at the time of writing this paper the best known general purpose algorithm for counting solutions to binary CSPs has a time complexity ranging from $\mathcal{O}\left((d\alpha^4/4)^n\right)$ to $\mathcal{O}\left((\alpha^5 + \alpha + \lfloor d/4 - 1 \rfloor \alpha^4)^n\right)$, where n is the number of variables, d is the size of the largest domain, and $\alpha \approx 1.2561$ is a constant [Angelsmark and Jonsson, 2003]. The problem of counting solutions to CSPs is also known to be intractable in general [Bulatov and Dalmau, 2003]. This is our main motivation for improving upon enumeration based techniques. Intractability does not invalidate our motivation for improving upon search based approaches for counting solutions because intractability poses a limit on the practicality of *all* algorithms—including the best, whereas having many solutions poses a limit on the practicality of *search based* algorithms.

The third and fourth algorithm overcome some of the weaknesses of search based algorithms. They eliminate a class of permutations acting upon the *entire* label set of a given partial order $\mathcal{P}$. The permutations they eliminate are permutations between the label sets of disconnected components of $\mathcal{P}$. They reduce the frequency of $\mathcal{P}$ to the product of a multinomial coefficient and the product of the frequencies of its components. The fourth algorithm also eliminates a class of permutations acting upon *subsets* of the label set of $\mathcal{P}$. The difference between the kinds of symmetry which the algorithms eliminate can roughly be characterised as follows. Both algorithms avoid labelling the same suborder with *different* label sets. The fourth algorithm also avoids repeatedly labelling the same suborder with *the same* label set. For moderately sized problems the techniques presented in this paper reduce the total solution time from several hours with MAC to 0.01 second and from more time than the currently estimated age of our universe to within a second.

We briefly discuss extensions of the techniques presented in this paper to different kinds of problems including permutation problems.

The remainder of this paper is organised as follows. Section 2 presents the mathematical background. This is followed by Section 3 which is an introduction to constraint satisfaction theory. Section 4 demonstrates how to encode the problem of computing the frequency of a partial order as an enumeration CSP. This is followed by Section 5 which presents the two improvements to the algorithms presented in Section 4. Section 6 studies extensions of the techniques for counting solutions to different kinds of problems including bijection problems. Experimental results are presented in Section 7. Section 8 presents conclusions and discusses future work.

2 Mathematical Background

A *partial order* is an ordered pair $\langle N, \sqsubseteq \rangle$, where N is a set and $\sqsubseteq$ is a reflexive and transitive relation on N such that $v \sqsubseteq w \wedge w \sqsubseteq v \Longrightarrow v = w$ for all $v, w \in N$. We will write $v \sqsubset w$ for $v \sqsubseteq w \wedge v \neq w$ and $w \sqsupseteq v$ for $v \sqsubseteq w$.

Example 1. The pair $\langle \mathbb{N}, \leq \rangle$ is a partial order but the pair $\langle \mathbb{N}, \neq \rangle$ is not a partial order (because, for example $1 \neq 2 \wedge 2 \neq 1$ but it is not true that $1 = 2$).

Definition 1 (Labelling). *Let N be a finite set and let $\mathcal{P} = \langle N, \sqsubseteq \rangle$ be a partial order. A bijection $l : N \mapsto \mathcal{L}$ is called a* labelling *of $\mathcal{P}$ if $v \sqsubset w \Longrightarrow l(v) < l(w)$, for all $v, w \in N$. If in addition $\mathcal{L} = \{ 1, \ldots, \mid N \mid \}$ then it is called a* standard *labelling of $\mathcal{P}$.*

Example 2. Let $N = \{ a, b, c \}$ and let $R = \{ \langle a, a \rangle, \langle b, b \rangle, \langle c, c \rangle, \langle a, b \rangle, \langle a, c \rangle \}$. Then $\langle N, R \rangle$ has 2 standard labellings $l_1(\cdot)$ and $l_2(\cdot)$, which are defined as follows:

$$l_1(a) = 1\,, \quad l_1(b) = 2\,, \quad l_1(c) = 3\,; \quad l_2(a) = 1\,, \quad l_2(b) = 3\,, \quad l_2(c) = 2\,.$$

Definition 2 (Frequency). *Let N be a finite set and let $\mathcal{P} = \langle N, \sqsubseteq \rangle$ be a partial order. The* frequency *of $\mathcal{P}$, denoted $\mathrm{f}(\mathcal{P})$, is the number of standard labellings of $\mathcal{P}$.*

Example 3. Let $\mathcal{P} = \langle N, R \rangle$ be the partial order from Example 2. Then $\mathcal{P}$ has only 2 standard labellings. Therefore, $\mathrm{f}(\mathcal{P}) = 2$.

Example 4. Let N be a finite set, then $\mathrm{f}(\langle N, \{ \langle n, n \rangle : n \in N \} \rangle)$ is $\mid N \mid !$.

Let $\mathcal{P} = \langle N, \sqsubseteq \rangle$ be a partial order. Throughout this paper we will refer to N as the *nodes* of $\mathcal{P}$ and to $\{ 1, \ldots, \mid N \mid \}$ as the *standard label set* of $\mathcal{P}$.

This paper is concerned with the following question.

Question 1 (Frequency). What is the frequency of a given partial order.

3 Constraint Satisfaction

A *constraint* is an ordered tuple $\langle S, R \rangle$, where S is an ordered sequence of variables called the *scope* of the constraint and R is a subset of the Cartesian product of the domains of the variables in S. The domain of variable x is denoted $D(x)$. The *arity* of a constraint is the length of its scope. A constraint is called *binary* if its arity is equal to 2. A constraint $\langle \langle x_{i_1}, \ldots, x_{i_m} \rangle, R \rangle$ is called *hyper-arc consistent* if for all $1 \leq k \leq m$ and for all $v_k \in D(x_{i_k})$ it holds that $\{ \langle w_1, \ldots, w_m \rangle \in R : v_k = w_k \} \neq \emptyset$. A hyper-arc consistent constraint is called *arc consistent* if it is binary.

A *Constraint Satisfaction Problem* (CSP) is a tuple $\langle X, D, C \rangle$, where X is a set of variables, $D(\cdot)$ is a function mapping each $x \in X$ to its non-empty domain, and C is a set of constraints among variables in subsets of X. A CSP is called *binary* if its constraints are binary. A CSP (binary CSP) is called *hyper-arc consistent* (*arc consistent*) if its constraints are hyper-arc consistent. A *solution* to CSP $\langle X, D, C \rangle$ is a function $\mathcal{S} : X \mapsto \cup_{x \in X} D(x)$ such that

$$(\forall \langle \langle x_{i_1}, \ldots, x_{i_m} \rangle, R \rangle \in C)(\langle \mathcal{S}(x_{i_1}), \ldots, \mathcal{S}(x_{i_m}) \rangle \in R)\,.$$

Two CSPs are *equivalent* if they have the same solutions. To transform a CSP to its hyper-arc consistent equivalent does not remove or add solutions and usually significantly reduces the search space. Arc consistency can be established within an asymptotic worst case time-complexity that is polynomial in the size of the CSP [Mackworth and Freuder, 1985]. The overhead of maintaining arc consistency during backtrack search significantly improves the average time-complexity of backtracking algorithms [Sabin and Freuder, 1994].

4 Formulating the Labelling Problem as a CSP

This section presents a CSP formulation for computing standard labellings of a given partial order. This provides an immediate implementation for computing the frequency of that partial order by counting all solutions of the CSP that is associated with that partial order.

Let $\langle X, \sqsubseteq \rangle$ be a partial order. Furthermore, let $D(x) = \{1, \ldots, |X|\}$, for $x \in X$, and let C be given by

$$C = \left\{ \langle \langle v, w \rangle, R_{vw} \rangle : \langle v, w \rangle \in X^2 \wedge v < w \right\},$$

where

$$R_{vw} = \begin{cases} \{ \langle i, j \rangle \in D(v) \times D(w) : i \neq j \} & \text{if } \neg(v \sqsubseteq w) \wedge \neg(w \sqsubseteq v), \\ \{ \langle i, j \rangle \in D(v) \times D(w) : i < j \} & \text{if } v \sqsubset w, \\ \{ \langle i, j \rangle \in D(v) \times D(w) : j < i \} & \text{if } w \sqsubset v. \end{cases}$$

It is left as an exercise to the reader to prove that $\langle X, D, C \rangle$ is a CSP whose solutions are exactly the standard labellings of $\langle X, \sqsubseteq \rangle$.

An alternative formulation is to omit all disequality constraints $\cdot \neq \cdot$ and instead post a global `alldifferent` constraint on the variables of the CSP. This constraint forces all pairs of variables to be pairwise different. The advantage of this formulation is that efficient algorithms are known for enforcing hyper-arc consistency for this global constraint [Régin, 1994] (see also [van Hoeve, 2001] for a survey of the `alldifferent` constraint). An arc consistent CSP using a binary encoding of the `alldifferent` constraint is not always equivalent to a hyper-arc consistent CSP with an `alldifferent` constraint.

Figure 1 depicts the *Hasse diagram* of a partial order. An arc in a Hasse diagram that goes up from w to v indicates that $v \sqsubset w$. Hasse diagrams of partial orders omit loops following from reflexivity and arcs following from transitivity. This is why, for example, the arc $\langle 1, 6 \rangle$ is not depicted in Figure 1 (because it follows from the transitivity of the arcs $\langle 1, 4 \rangle$ and $\langle 4, 6 \rangle$).

Example 5. The code in Figure 2 is an implementation in the ECL^{i}PS^e [Wallace *et al.*, 1997] constraint programming language for computing the frequency of the order depicted in Figure 1. ECL^{i}PS^e's implementation of the `alldifferent` constraint does not use Régin's hyper-arc consistency algorithm [Régin, 1994] but uses a collection of binary disequality constraints for ensuring that each variable takes a different value.

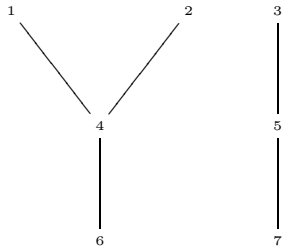

Fig. 1. A disconnected order.

```
count_order( FREQUENCY ) :-
  constrain( VARS ),
  countsols( VARS, FREQUENCY ).

constrain( VARS ) :-
  VARS = [X1,X2,X3,X4,X5,X6,X7],
  VARS :: 1..7,
  alldifferent( VARS ),
  X1 #< X4, X2 #< X4, X4 #< X6,
  X3 #< X5, X5 #< X7.

countsols( VARS, NRSOLS ) :-
  setval( sols, 0 ),
  countsols( VARS ).
  getval( sols, NRSOLS ).
countsols( XS ) :-
  labeling( XS ),
  incval( sols ),
  fail.
countsols( _ ).
```

Fig. 2. ECL^{i}PS^e code for enumerating disconnected order.

We used ECL^{i}PS^e for computing the frequency of many different orders. Also we used OPL [Van Hentenryck, 1999] and the MAC-3_d implementation described in [van Dongen, 2004]. The ECL^{i}PS^e and MAC-3_d implementations use binary disequality constraints for enforcing a standard labelling of the problem. The OPL implementation uses Régin's filtering constraint [Régin, 1994]. The ECL^{i}PS^e and OPL versions were about equally efficient in time. However, the MAC-3_d version was about ten times faster. Our CSP-based implementations improved on dedicated programs for frequency computation that were previously developed at CEOL.

5 Counting by Removing One or Several Nodes

As demonstrated by Example 4 there are partial orders whose frequencies are $n!$, where n is the number of nodes in the partial order. As laid out in the introduction this poses problems for algorithms that count by enumerating each single solution.

This section describes two algorithms which improve on enumeration based algorithms. Both algorithms assign labels to nodes. The first algorithm always assigns the largest or the smallest label to the nodes that can be assigned that label. The second algorithm is a generalisation. It always assigns the s largest or s smallest labels to the subsets of nodes that can be assigned those s labels. Their labelling strategy allows the algorithms to remove nodes from the partial order that were already labelled and this may lead to disconnected components, with which they can deal efficiently.

Both improved algorithms exploit two combinatorial properties of partial orders for counting efficiently. Both algorithms eliminate all permutations between the labels of disconnected components of a given partial order $\mathcal{P}$. These permutations occur in the *entire* label set of $\mathcal{P}$. This improvement avoids labelling components of partial orders with *different* label sets. The second improved algorithm also eliminates permutations occurring in *subsets* of the label set of $\mathcal{P}$. This improvement avoids multiple labellings of a subset of the nodes of $\mathcal{P}$ with *the same* label set.

Before presenting the improvements we need some notation for describing and reasoning about "restrictions" of partial orders to subsets of its nodes and about "connected components" of partial orders.

Definition 3 (Restriction). *The* restriction *of* $\langle N, R \rangle$ *to* $M \subseteq N$ *is defined as* $\langle M, R \cap M^2 \rangle$.

Example 6. The restriction of the order whose Hasse diagram is depicted in Figure 1 to $\{ 1, 2, 4, 6 \}$ corresponds to the leftmost component in that Hasse diagram.

Definition 4 (Connected Component). *Let* $\mathcal{P} = \langle N, \sqsubseteq \rangle$ *be a partial order and let* $N' \subseteq N$. *The restriction* $\langle N', \sqsubseteq \rangle$ *of* $\mathcal{P}$ *to* N' *is called a* (connected) component *of* $\mathcal{P}$ *if each of the following is true:*

- *For all* $u, w \in N'$ *there exist a (possibly empty) set* $\{ v_1, \ldots, v_m \} \subseteq N'$ *and* $\preceq_i \in \{ \sqsubseteq, \sqsupseteq \}$, *for* $1 \leq i \leq m+1$, *such that* $u \preceq_1 v_1 \preceq_2 \cdots \preceq_m v_m \preceq_{m+1} w$.
- *For all* $u \in N'$ *and all* $w \in N \setminus N'$ *there do not exist a set* $\{ v_1, \ldots, v_m \} \subseteq N$ *and* $\preceq_i \in \{ \sqsubseteq, \sqsupseteq \}$, *for* $1 \leq i \leq m+1$, *such that* $u \preceq_1 v_1 \preceq_2 \cdots \preceq_m v_m \preceq_{m+1} w$.

The connected components of a given partial order correspond to the maximally connected components in the Hasse diagram of that order.

Example 7. Consider the order $\mathcal{P}$ which is depicted in Figure 1. It has two components $\mathcal{P}'$ and $\mathcal{P}''$, where $\mathcal{P}'$ is the restriction of $\mathcal{P}$ to $\{ 1, 2, 4, 6 \}$ and where $\mathcal{P}''$ is the restriction of $\mathcal{P}$ to $\{ 3, 5, 7 \}$. Since the nodes of $\mathcal{P}'$ and the nodes of $\mathcal{P}''$ are not related in $\mathcal{P}$ we can label the orders $\mathcal{P}'$ and $\mathcal{P}''$ independently *provided* we make sure that their label sets are disjoint. Indeed, we can label $\mathcal{P}'$ with *any* four labels from $\mathcal{L} = \{ 1, 2, 3, 4, 5, 6, 7 \}$, label $\mathcal{P}''$ with the remaining three nodes and combine the two resulting labellings to get a standard labelling of $\mathcal{P}$. All labellings of $\mathcal{P}'$ with any 4 labels selected from $\mathcal{L}$ are isomorphic to the 2 standard labellings of $\mathcal{P}'$. For example, the standard labellings of $\mathcal{P}'$ are $\{ 1 \mapsto 1, 2 \mapsto 2, 4 \mapsto 3, 6 \mapsto 4 \}$ and $\{ 1 \mapsto 2, 2 \mapsto 1, 4 \mapsto 3, 6 \mapsto 4 \}$, and the labellings of $\mathcal{P}'$ with $l_1 < l_2 < l_3 < l_4$ are given by $\{ 1 \mapsto l_1, 2 \mapsto l_2, 4 \mapsto l_3, 6 \mapsto l_4 \}$ and $\{ 1 \mapsto l_2, 2 \mapsto l_1, 4 \mapsto l_3, 6 \mapsto l_4 \}$. Similarly, all labellings of $\mathcal{P}''$ with any three labels from $\mathcal{L}$ are isomorphic to the single standard labelling of $\mathcal{P}''$. An intelligent algorithm ought to exploit the fact that there are exactly $\binom{7}{4} = 35$ ways to select 4 labels from the 7 labels in $\mathcal{L}$, that there is exactly $\binom{3}{3} = 1$ way to select 3 labels from the remaining 3 labels, that the labellings of $\mathcal{P}'$ with 4 labels are isomorphic to the 2 standard labellings of $\mathcal{P}'$, that the labellings of $\mathcal{P}''$ with the three remaining labels are isomorphic to the 1 standard labelling of $\mathcal{P}''$, and that therefore the number of standard labellings of $\mathcal{P}$ is equal to $35 \times 1 \times 2 \times 1 = 70$.

The *multinomial coefficient* of $n_1, \ldots, n_m$ is the number of different ways to partition a set having a Cardinality of $n_1 + \cdots + n_m$ into m disjoint sets having Cardinalities $n_1, \ldots, n_m$. The multinomial coefficient of $n_1, \ldots, n_m$ is denoted $\binom{n_1+\cdots+n_m}{n_1,\ldots,n_m}$. It is not difficult to show that $\binom{n_1+\cdots+n_m}{n_1,\ldots,n_m} = \frac{(n_1+\cdots+n_m)!}{(n_1!)\times\cdots\times(n_m!)}$. For the case where $m = 2$ the multinomial coefficient reduces to the binomial coefficients $\binom{n_1+n_2}{n_1} = \binom{n_1+n_2}{n_2}$.

The following theorem states a relationship between the frequency of a given partial order, the multinomial coefficient of the sizes of its connected components and the frequencies of its connected components.

Theorem 1 (Multinomial Property). *Let $\mathcal{P} = \langle N, R \rangle$ be a partial order and let N_1, ..., N_m be pairwise disjoint non-empty sets such that $\langle N, R \rangle = \langle \cup_{i=1}^{m} N_i, \cup_{i=1}^{m} R \cap N_i^2 \rangle$, then*

$$\mathrm{f}(\mathcal{P}) = \binom{|N_1| + \cdots + |N_m|}{|N_1|, \ldots, |N_m|} \times \prod_{i=1}^{m} \mathrm{f}(\langle N_i, R \cap N_i^2 \rangle) .$$

Proof. There are $\binom{|N_1|+\cdots+|N_m|}{|N_1|,\ldots,|N_m|}$ different ways of partitioning $\{1, \ldots, |N|\}$ into label sets $L_1, \ldots, L_m$, such that $L_i \cap L_j = \emptyset \iff i \neq j$ and such that $|L_i| = |N_i|$, for $1 \leq i, j \leq m$. Each partition corresponds to $\prod_{i=1}^{m} l(N_i, L_i)$ labellings of $\mathcal{P}$, where $l(N_i, L_i)$ is the number of labellings of $\langle N_i, E \cap N_i^2 \rangle$ with L_i. For each i such that $1 \leq i \leq m$ the standard labellings of N_i are isomorphic to the labellings of N_i with L_i. Therefore, we have $l(N_i, L_i) = \mathrm{f}(\langle N_i, E \cap N_i^2 \rangle)$, for $1 \leq i \leq m$, and it follows that each partition corresponds to $\prod_{i=1}^{m} \mathrm{f}(\langle N_i, E \cap N_i^2 \rangle)$ standard labellings of $\mathcal{P}$.

Definition 5 (Minima/Maxima). *Let $\mathcal{P} = \langle N, R \rangle$ be a partial order. The* minima *of $\mathcal{P}$ are denoted* $\mathrm{minima}(\mathcal{P})$ *and the* maxima *of $\mathcal{P}$ are denoted* $\mathrm{maxima}(\mathcal{P})$*. The minima and maxima of $\mathcal{P}$ are defined as*

$$\mathrm{minima}(\mathcal{P}) = \{ v \in N : (\{v\} \times N) \cap R = \{ \langle v, v \rangle \} \} ,$$

and

$$\mathrm{maxima}(\mathcal{P}) = \{ v \in N : (N \times \{v\}) \cap R = \{ \langle v, v \rangle \} \} .$$

Example 8. The minima of the partial order depicted in Figure 1 are given by $\{1, 2, 3\}$ and the maxima of this order are given by $\{6, 7\}$.

The multinomial property lets us reduce a problem having *several* connected components to several smaller labelling problems. The following lemma allows us to reduce a problem having only *one* connected component to a smaller problem.

Lemma 1. *Let $\mathcal{P} = \langle N, \sqsubseteq \rangle$ be a partial order and let $m \in N$ then there exists a standard labelling $f(\cdot)$ of $\mathcal{P}$ such that $f(m) = |N|$ ($f(m) = 1$) if and only if $m \in \mathrm{maxima}(\mathcal{P})$ ($m \in \mathrm{minima}(\mathcal{P})$).*

Proof. First assume that $m \notin \mathrm{maxima}(\mathcal{P})$ and that $f(\cdot)$ is a standard labelling of $\mathcal{P}$ such that $f(m) = n = |N|$. Since $m \notin \mathrm{maxima}(\mathcal{P})$ there exists some $m' \in N$ such that $m \sqsubset m'$ and $n = f(m) < f(m') \leq n$, which is a contradiction. Next assume $m \in \mathrm{maxima}(\mathcal{P})$. Let $\mathcal{P}'$ be the restriction of $\mathcal{P}$ to $N \setminus \{m\}$ and let $g(\cdot)$ be a standard labelling of $\mathcal{P}'$. Define $f(m') = g(m')$ if $m' \neq m$ and $f(m) = n$, then $f(\cdot)$ is a bijection from N to $\{1, \ldots, n\}$. Furthermore, we have $f(v) = g(v) < g(w) = f(w)$ for all $v, w \in N \setminus \{m\}$ such that $v \sqsubset w$. Finally, we have $f(v) < n = f(m)$ for all $v \notin \mathrm{maxima}(\mathcal{P})$, and as a consequence we also have $f(m') < f(m)$ for all $m' \in N$ such that $m' \sqsubset m$. Therefore, $f(\cdot)$ is a standard labelling of $\mathcal{P}$. The proof for $m \in \mathrm{minima}(\mathcal{P})$ and $f(m) = 1$ is analogous.

The following theorem provides the foundation for our first improved algorithm.

```
Function frequency(Partial Order of Integer ⟨ N, E ⟩): Integer;
   Integer product, sum;
Begin
   product := | N |!;
   Foreach ⟨ N', E' ⟩ In connected_components(⟨ N, E ⟩) Do
      If | N' | > 1 Then Begin
         sum := 0;
         Foreach m ∈ minima(⟨ N', E' ⟩) Do
            sum := sum + frequency(⟨ N' \ { m } , E' ∩ (N' \ { m })^2 ⟩);
         product := product × sum/| N' |!;
      End;
   Return product;
End;
```

Fig. 3. Algorithm for computing the frequency of a partial order.

Theorem 2. *Let $\mathcal{P} = \langle N, R \rangle$ be a partial order and let $M = \mathrm{maxima}(\mathcal{P})$, then*

$$\mathrm{f}(\mathcal{P}) = \sum_{m \in M} \mathrm{f}(\langle N \setminus \{ m \}, R \cap (N \setminus \{ m \})^2 \rangle) .$$

Proof. By Lemma 1 only the maxima can be assigned the largest label. The number of standard labellings of $\mathcal{P}$ where a given maximal node m is labelled with the largest label is equal to the number of standard labellings of the restriction of $\mathcal{P}$ to $N \setminus \{ m \}$.

Note that Theorem 2 remains true when substituting minima for maxima. Theorems 1 and 2 suggest an algorithm for computing frequencies of partial orders. Pseudo-code for this algorithm is depicted in Figure 3. The function uses a predefined function called $connected\_components(\cdot)$ for computing the connected components of the partial order. It is left as an exercise to the reader to prove that the algorithm is correct.

The *frequency* algorithm depicted in Figure 3 significantly improves upon the algorithms described in Section 4. It maintains global consistency and exploits the multinomial property to avoid labelling the same partial order with different label sets.

However, the following example demonstrates that despite these improvements it still suffers from a serious problem because it frequently (implicitly) labels the same partial order with the same label set.

Example 9. Consider the partial order $\mathcal{P}$, which is depicted in Figure 1. Let $\mathcal{P}'$ be the restriction of $\mathcal{P}$ to $\{ 1, 2 \}$ and let $\mathcal{P}''$ be the restriction of $\mathcal{P}$ to $\{ 3, 4, 5, 6, 7 \}$. As part of the work for computing $\mathrm{f}(\mathcal{P})$ the *frequency* algorithm has to compute all the labellings whose restrictions to $\mathcal{P}'$ are standard labellings. To compute this subset of labellings it first computes the labelling $\{ 1 \mapsto 1, 2 \mapsto 2 \}$ by first removing node 1 and then node 2, then computes $\mathrm{f}(\mathcal{P}'')$, then computes the labelling $\{ 1 \mapsto 2, 2 \mapsto 1 \}$ by first removing node 2 and then node 1, and finally computes $\mathrm{f}(\mathcal{P}'')$. It computes $\mathrm{f}(\mathcal{P}'')$ more than once and does more work than is needed. The *reason* why the algorithm is doing too much work is that it labels by removing nodes and that the *order* in which the nodes of $\mathcal{P}'$ are labelled with the first labels makes no difference to the number of labellings of $\mathcal{P}''$ with the remaining labels. We can avoid recomputing the frequency of $\mathcal{P}''$ by observing that the total number of standard labellings of $\mathcal{P}$ where $\mathcal{P}'$ are labelled with the first labels is equal to $\mathrm{f}(\mathcal{P}') \times \mathrm{f}(\mathcal{P}'')$.

This suggests another improvement to our algorithm. Before presenting the improvement we need some terminology.

Definition 6. *Let $\mathcal{P} = \langle N, R \rangle$ be a partial order and let S be a non-empty subset of N. Then S is called* maximal *with respect to $\mathcal{P}$ if there exists a standard labelling $l(\cdot)$ of $\mathcal{P}$ such that $\{ l(s) : s \in S \} = \{ |N| - |S| + 1, \ldots, |N| \}$. S is called* minimal *with respect to $\mathcal{P}$ if there exists a standard labelling $l(\cdot)$ of $\mathcal{P}$ such that $\{ l(s) : s \in S \} = \{ 1, \ldots, |S| \}$.*

Example 10. Consider the partial order $\mathcal{P} = \langle N, R \rangle$, where $N = \{ a, b, c, d \}$ and $R = \{ \langle a, a \rangle, \langle b, b \rangle, \langle c, c \rangle, \langle d, d \rangle, \langle a, b \rangle, \langle b, d \rangle, \langle c, d \rangle \}$. The standard labellings of $\mathcal{P}$ are given by $\{ a \mapsto 2, b \mapsto 3, c \mapsto 1, d \mapsto 4 \}$, by $\{ a \mapsto 1, b \mapsto 3, c \mapsto 2, d \mapsto 4 \}$, and by $\{ a \mapsto 1, b \mapsto 2, c \mapsto 3, d \mapsto 4 \}$. The minimal sets of $\mathcal{P}$ are given by $\{ a \}$, $\{ c \}$, $\{ a, b \}$, $\{ a, c \}$, $\{ a, b, c \}$, and $\{ a, b, c, d \}$. The maximal sets of $\mathcal{P}$ are given by $\{ d \}$, $\{ b, d \}$, $\{ c, d \}$, $\{ b, c, d \}$, $\{ a, b, d \}$, and $\{ a, b, c, d \}$.

Theorem 3. *Let $P = \langle N, R \rangle$ be a partial order and let $1 \leq s \leq |N|$, then*

$$\mathrm{f}(P) = \sum_{\substack{S \subseteq N \\ |S| = s \\ S \text{ is maximal}}} \mathrm{f}(\langle N \setminus S, R \cap (N \setminus S)^2 \rangle) \times \mathrm{f}(\langle S, R \cap S^2 \rangle) .$$

Proof. By induction on s and application of Theorem 2.

Note that Theorem 3 also holds if the word minimal is substituted for maximal.

```
Function frequency'(Integer size, Partial Order of Integer ⟨ N, E ⟩): Integer;
   Integer product;
   Function freq(Integer size, Partial Order of Integer ⟨ N, E ⟩): Integer;
      Integer sum;
      Set of Integer D;
   Begin
      If |N| ≤ 1 Then
         Return 1;
      Else If size ≥ |N| Then
         Return freq(size/2, ⟨ N, E ⟩);
      Else Begin
         sum := 0;
         Foreach M ∈ minimal_sets(size, ⟨ N, E ⟩) Do Begin
            D := N \ M;
            sum := sum + frequency'(size, ⟨ M, E ∩ M² ⟩) × frequency'(size, ⟨ D, E ∩ D² ⟩);
         End;
         Return sum;
      End;
   End;
Begin
   product := |N|!;
   Foreach ⟨ N', E' ⟩ In connected_components(⟨ N, E ⟩) Do
      product := product × freq(size, ⟨ N', E' ⟩)/|N'|!;
   Return product;
End;
```

Fig. 4. Improved algorithm for computing the frequency of partial orders.

The algorithm called $frequency'$, which is depicted in Figure 4, uses Theorem 3 to compute the frequency of partial orders. The algorithm relies on a predefined function $minimal\_sets(size, \mathcal{P})$ for computing the minimal sets of $\mathcal{P}$ that have a Cardinality that is equal to $size$. The parameter $size$ is called the *size-parameter* of the algorithm. It corresponds to the integer s in Theorem 3. It is straightforward to prove termination and correctness of the algorithm.

6 Extending the Techniques

This section briefly discusses how to generalise the techniques for counting other kinds of bijection problems, i.e. problems where the solutions are bijections between the variables and the values of that problem. Note that some problems that are not bijection problems can be transformed to bijection problems if it is known that a certain value v in a solution always occurs m times by replacing v by fresh values $v_1, \ldots, v_m$ and by adding constraints that no two variables can take the same value v_i, for $1 \leq i \leq m$. In that sense the generalisation presented in this section applies to a class of problems including bijection problems.

For the purpose of this section let a *simple* subproblem $\langle X', D', C' \rangle$ of problem $\langle X, D, C \rangle$ be a problem such that whenever there is a constraint in C among non-empty $T \subseteq X \setminus X'$ and non-empty $T' \subseteq X'$ then it is of the form alldifferent$(T \cup T')$. Furthermore, let a subproblem $\mathcal{P}'$ of $\mathcal{P}$ be called a *maximal* simple subproblem of $\mathcal{P}$ if there are no simple subproblems of $\mathcal{P}$ properly including $\mathcal{P}'$. If the Multinomial Theorem (Theorem 1) applies then we can count maximal simple subproblems independently.

Consider any bijection problem $\mathcal{P}$. If there is only one variable then there is exactly one solution. Assume there is more than one variable. Let P be any non-trivial property that holds for at least one variable but not for all. Properties of variables correspond to relations of unary constraints, i.e. to sets of values. Our restrictions dictate that P be a non-trivial subset of the union of the domains of the variables. Because our problems are bijection problems the property P partitions the variables into two non-empty subsets: the variables than can be assigned the values in P and the variables that cannot be assigned the values in P. Note that it is always possible to compute a non-trivial property from the union of the domains of the variables.

The Multinomial Property allows us to solve maximal simple subproblems independently. Assume that $\mathcal{P}$ is a maximal simple subproblem of itself. Remember that P may be regarded as a constraint, i.e. a set of values. If we have a cheap algorithm for determining whether the partial assignment $x := v \in P$ can be extended to a solution then we have all that is needed count efficiently. We pick any value $v \in P$ and for each variable x for which the cheap algorithm tells us that the assignment $x := v$ can be successfully extended to a solution we compute solutions the problem where x is removed from the variables and v from the domains of the variables. This is essentially the equivalent of Theorem 2. By taking subsets of P of arbitrary size we can also apply the equivalent of Theorem 3.

In Theorem 2 $\mathrm{minima}(\langle\, \mathcal{P} \,\rangle)$ plays the role of the variables of $\mathcal{P}$ that can be assigned the value 1. In terms of the description in this section it may be regarded as the variables of $\mathcal{P}$ satisfying the property $P = \{\, 1 \,\}$.

7 Experimental Results

This section presents results from an experimental comparison of the algorithms described in the previous sections. All results were obtained on a 1000 MHz DELL Latitude.

In the following let $K_{m,n}$ denote the *maximal bi-partite order*, which is defined as

$$K_{m,n} = \langle\, \{\, 1, \ldots, m+n \,\}, \sqsubseteq \,\rangle\,,$$

where $i \sqsubset j \iff i \leq m < j$. Suborders of $K_{m,n}$ occur frequently as suborders of partial orders. Such suborders are the among the most difficult orders to count for *frequency* and *frequency*$'$. They are difficult to count because it is difficult to recognise them as parts of problems (because subgraph-isomorphism is a difficult problem). Note that the frequency of $K_{m,n}$ is easy to compute for humans—it is $(m!) \times (n!)$. Considering connected orders only, "tree shaped orders" are among the easiest ones to count. Assuming that all arithmetic operations have a constant time complexity, algorithms *frequency* and *frequency*$'$ can compute the frequency of trees in a time which is polynomial in the number of nodes by repeatedly removing the roots from the trees. Counting by removing leaves is more difficult. Let B_n^+ (B_n^-) denote the partial order whose Hasse diagram corresponds to the complete binary tree with $2^{1+n} - 1$ nodes, that is rooted at the bottom (top). Note that B_n^+ and B_n^- have the same frequency for all n. However, the labellings of B_n^+ are more difficult to count for *frequency*$'$ because it only removes minimal sets. The sequence $\langle\, \mathrm{f}(B_n^+) \,\rangle_{n \geq 0}$ is known as Sloane's Sequence A056972 (`http://www.research.att.com/cgi-bin/access.cgi/as/njas/sequences/eisA.cgi?Anum=A056972`).

We implemented *frequency* and *frequency*$'$ in the `Prolog` subset of ECL^{i}PS^e. Excluding predicates for graph manipulation, the total code for *frequency*$'$ in ECL^{i}PS^e required a single A4 page of code.

The improvements of *frequency* and *frequency*$'$ over the MAC-based algorithms are significant. As mentioned before, our best MAC-based algorithm is MAC-3_d. It was about ten times more efficient in time then the ECL^{i}PS^e and OPL implementations. MAC-3_d, required more than 6 hours for computing the frequency of $K_{8,8}$. An implementation with OPL did not terminate after many hours and required an intermediate memory size of more than 60MB. These findings are in sharp contrast with *frequency*$'$ which required only 0.01 seconds for computing $\mathrm{f}(K_{8,8})$ with a size-parameter of 8 and fewer than 2 seconds with a size-parameter of 2.

Table 1 lists the results of applying algorithm *frequency*$'$ with different size-parameters. The first column in this table describes the problem, the second column lists the size of the size-parameter, the third column lists the frequency of the order, and the last column lists the time which was required to compute the frequency. The results from this table for a size equal to 1 are slightly worse than the results for the *frequency* algorithm. We expect that a proper implementation of *frequency*$'$ will avoid the stack overflows from which it is currently suffering for small size-parameters.

The results for B_n^- with a size-parameter of 1 demonstrate the advantage of using Theorem 1 because each time a node is removed from an order having more than one node this results in two disconnected components of equal size. The results for the $K_{n,n}$ and B_n^+ demonstrate the advantage of Theorem 3 because as the size-parameter increases for a given problem less and less time is required.

Problem	Size	Frequency	Time in Seconds
B_1^+	1	2	0.00
B_1^-	1	2	0.00
B_2^+	1	80	0.00
B_2^-	1	80	0.00
B_3^+	1–3	stack overflow	—
B_3^+	4	21964800	3.45
B_3^-	1	21964800	0.00
B_4^-	1	74836825861835980800000	0.01
B_5^-	1	$10^{63.42}$	0.03
B_6^-	1	$10^{163.61}$	0.21
B_7^-	1	$10^{4023.86}$	0.93
B_8^-	1	$10^{9568.46}$	3.96
$K_{8,8}$	1	stack overflow	—
$K_{8,8}$	2	1625702400	1.96
$K_{8,8}$	4	1625702400	0.09
$K_{8,8}$	8	1625702400	0.01
$K_{10,10}$	1–4	stack overflow	—
$K_{10,10}$	5	13168189440000	0.66
$K_{10,10}$	10	13168189440000	0.06
$K_{12,12}$	1–5	stack overflow	—
$K_{12,12}$	6	229442532802560000	5.42
$K_{12,12}$	12	229442532802560000	0.38
$K_{14,14}$	14	7600054456551997440000	1.92
$K_{16,16}$	16	437763136697395052544000000	9.30

Table 1. Timing results for $frequency'$ algorithm.

Assuming that the MAC-based algorithms count one standard labelling per clock cycle on our test machine, they would require more than 10^{147} years for computing the frequency of B_6^-, which is more time than the estimated age of our universe, which is currently being estimated to be 13–14 billion years old. However, $frequency'$ computes the frequency of B_6^- in less than 0.3 seconds, and this demonstrates the strength of the techniques presented in this paper.

We also applied the algorithms to other orders not corresponding to bipartite graphs or complete trees. For these orders we also observed that the $fequency$ and $frequency'$ algorithms were superior to the other algorithms.

The current implementation of $frequency'$ is not particularly clever at exploiting structural properties of the partial orders whose frequency it is computing. This is especially demonstrated by the differences in time for computing the frequencies of B_n^+

and B_n^-, each of which having the same frequency. The reason for the difference in time is that currently the algorithm only removes minimal sets. It should be possible to improve the algorithm by allowing it to also remove maximal sets. With a proper heuristic for deciding which of the two kinds of sets to pick this should improve the algorithm. Another weakness of the algorithm is that it does not take symmetries of the constraint graph into account. A final improvement to $frequency'$ is to more tightly integrate the computation of the minimal sets and the counting, i.e. to start a recursive count immediately after the computation of the next minimal set. This will avoid the creation of potentially large intermediate lists for storing the minimal sets.

8 Conclusions

We studied four algorithms for computing the frequency of a given partial order. The first two algorithms are based on the correspondence between partial orders and constraint satisfaction problems (CSPs). They use backtrack search while maintaining arc consistency to enumerate and count all solutions. A disadvantage of these algorithms is that they soon become infeasible due to there being many solutions.

The third and fourth algorithm overcome some of the weaknesses of the search based algorithms. They eliminate a class of permutations acting upon the *entire* label set of a given partial order $\mathcal{P}$. The permutations they eliminate are permutations between the label sets of disconnected components of $\mathcal{P}$. This reduces the frequency of $\mathcal{P}$ to the product of a multinomial coefficient and the product of the frequencies of its components. The fourth algorithm also eliminates a class of permutations that are restricted to *subsets* of the label set of $\mathcal{P}$. For moderately sized problems the techniques presented in this paper reduce the total solution time from several hours with MAC to 0.01 second and from more time than the currently estimated age of our universe to within a second.

The most promising improvement to the fourth algorithm is to allow it to actively search—but not too hard—for cut-sets whose removal will turn a given order into two or more unconnected components and to use these cut-sets to actively decompose the partial order, as opposed to waiting for this to happen. Another way is to improve its underlying representation for partial orders.

The techniques presented in this paper are very promising and they achieve impressive improvements over enumeration based algorithms. We discussed how to extend the techniques presented in this paper (or adaptations thereof) for counting solutions to bijection problems.

Acknowledgements

This work has received support from Science Foundation Ireland under Grant `02/IN.1/181`. The author wishes to thank the members of the Centre of Efficiency Orientated Languages and the Cork constraint Computation Centre. In particular he wishes to thank Nic Wilson and Yuan Lin Zhang for comments on an early draft.

References

[Angelsmark and Jonsson, 2003] O. Angelsmark and P. Jonsson. Improved algorithms for counting solutions in constraint satisfaction problems. In F. Rossi, editor, *Proceedings of the ninth International Conference on Principles and Practice of Constraint Programming (CP'2003)*, pages 81–95, 2003.

[Bulatov and Dalmau, 2003] A.A. Bulatov and V. Dalmau. Towards a dichotomy for the counting constraint satisfaction problem. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2003)*, pages 272–282, 2003.

[Mackworth and Freuder, 1985] A.K. Mackworth and E.C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25(1):65–73, 1985.

[Régin, 1994] Jean-Charles Régin. A filtering algorithm for constraints of difference in CSPs. In *Proceedings of the* 12^{th} *National Conference on Artificial Intelligence (AAAI'94)*, pages 362–367, 1994.

[Sabin and Freuder, 1994] D. Sabin and E.C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In A.G. Cohn, editor, *Proceedings of the* 11^{th} *European Conference on Artificial Intelligence (ECAI'94)*, pages 125–129. John Wiley and Sons, 1994.

[Schellekens, 2004] M.P. Schellekens. Compositional average time analysis *toward a calculus for software timing*. Technical report, Centre for Efficiency Orientated Languages, 2004. In Preparation.

[Sedgewick and Flajolet, 1996] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Publishing Company, 1996.

[van Dongen, 2004] M.R.C. van Dongen. Saving support-checks does not always save time. *Artificial Intelligence Review*, 2004. Accepted for publication.

[Van Hentenryck, 1999] P. Van Hentenryck. *The OPL Programming Language*. MIT Press, 1999.

[van Hoeve, 2001] W.J. van Hoeve. The alldifferent constraint: A survey. In *Sixth Annual Workshop of the ERCIM Working Group on Constraints*, 2001.

[Wallace *et al.*, 1997] M. Wallace, S. Novello, and j. Schimpf. ECL^{i}PS^e: A platform for constraint logic programming. Technical report, IC-Parc, Imperial College, 1997.