Computing the Frequency of Partial Orders

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

Centre for Efficiency Orientated Languages, University College Cork

Abstract. In this paper we 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. The first two algorithms count by enumerating all solutions to a CSP. 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 certain kinds of isomorphic solutions. A prototype implementation of the fourth algorithm was significantly more efficient than an enumeration based counting implementation using OPL.

1 Introduction

We study four algorithms for computing the *frequency* of partial orders. The problem of frequency computation is posed in [4], where it is called *counting labellings* (see also [5, Page 316]).

Our first two algorithms count by enumerating the solutions of a Constraint Satisfaction Problem (CSP). This soon becomes infeasible due to large numbers of solutions. At the time of writing 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 $\alpha \approx 1.2561$ [1]. Counting solutions to CSPs is known to be #P-complete in general [2]. The third and fourth algorithm overcome some of the weaknesses of search based algorithms. 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.

Section 2 presents mathematical background. Section 3 encodes frequency computation as enumeration CSPs. Section 4 presents two improvements to the CSP-based algorithms. Results are presented in Section 5. Section 6 presents conclusions.

2 Mathematical Background

A *partial order* is an ordered pair $\mathcal{P} = \langle N, \sqsubseteq \rangle$, where N is a set and $\sqsubseteq$ is a reflexive and transitive relation on N such that $v \sqsubseteq w \land w \sqsubseteq v \implies v = w$ for all $v, w \in N$. We will write $v \sqsubset w$ for $v \sqsubseteq w \land v \neq w$. A bijection $l : N \mapsto \mathcal{L}$ is called a *labelling* of $\mathcal{P}$

 This work has received support from Science Foundation Ireland under Grant 02/IN.1/181.

if $v \sqsubset w \Longrightarrow l(v) < l(w)$, for all $v, w \in N$. If in addition $\mathcal{L} = \{ 1, \ldots, |N| \}$ then l is called a *standard* labelling of $\mathcal{P}$. The *frequency* of $\mathcal{P}$, denoted $\mathrm{f}(\mathcal{P})$, is the number of standard labellings of $\mathcal{P}$.

3 Formulating the Labelling Problem as a CSP

A *constraint satisfaction problem* (CSP) is a tuple $\langle X, D, C \rangle$, where X is a set of variables, D is a function that maps each variable to its domain, and C is a set of constraints. Here a *constraint* is a pair $\langle S, R \rangle$, where S is an ordered sequence of variables and R is a subset of the Cartesian product of the variables in S. A *solution* of CSP $\langle X, D, C \rangle$ is a function s such that $s(x_1, \ldots, x_m) \in R$ for all $\langle \langle x_1, \ldots, x_m \rangle, R \rangle \in C$.

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 $C = \{ \langle \langle v, w \rangle, R_{vw} \rangle : \langle v, w \rangle \in X^2 \wedge v < w \}$, 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 replace all disequality constraints $\cdot \neq \cdot$ by a global `alldifferent` constraint. This has the advantage that efficient algorithms are known for enforcing hyper-arc consistency for this global constraint [3].

We implemented the algorithms using OPL [7] and MAC-3_d [6]. The MAC-3_d implementation uses binary disequality constraints, whereas the OPL implementation uses Régin's filtering constraint [3]. The MAC-3_d version was about ten times faster.

4 Counting by Removing One or Several Nodes

The *restriction* of $\mathcal{P} = \langle N, R \rangle$ to $M \subseteq N$ is defined as $\langle M, R \cap M^2 \rangle$. The restriction $\langle N', \sqsubseteq \rangle$ of $\mathcal{P}$ to $N' \subseteq N$ is called a *(connected) component* of $\mathcal{P}$ if both of the following are 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 $u \preceq_1 v_1 \preceq_2 \cdots \preceq_m v_m \preceq_{m+1} w$.

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

The *minima* of $\mathcal{P} = \langle N, R \rangle$, denoted $\mathrm{minima}(\mathcal{P})$, are defined as $\mathrm{minima}(\mathcal{P}) = \{ v \in N : (N \times \{ v \}) \cap R = \{ \langle v, v \rangle \} \}$. The *maxima* of $\mathcal{P}$, denoted $\mathrm{maxima}(\mathcal{P})$, are defined as $\mathrm{maxima}(\mathcal{P}) = \{ v \in N : (\{ v \} \times N) \cap R = \{ \langle v, v \rangle \} \}$.

```
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. 1. Algorithm for computing the frequency of a partial order.

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 2. *Let* $\mathcal{P} = \langle N, \sqsubseteq \rangle$ *and let* $m \in N$. $\mathcal{P}$ *has a standard labelling* $f(\cdot)$ *such that* $f(m) = |\, N \,|$ *(*$f(m) = 1$*) if and only if* $m \in \mathrm{maxima}(\mathcal{P})$ *(*$m \in \mathrm{minima}(\mathcal{P})$*).*

Theorem 3. *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 2 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 $N \setminus \{\, m \,\}$.

Theorem 3 remains true when substituting minima for maxima. Theorems 1 and 3 suggest an algorithm for computing frequencies of partial orders. Pseudo-code for this algorithm is depicted in Figure 1. It maintains global consistency and exploits the multinomial property to avoid labelling the same partial order with different label sets. Unfortunately, *frequency* frequently (implicitly) labels sub-orders with the same label set.

Let let $\emptyset \subset S \subseteq N$. S is *maximal* (*minimal*) with respect to $\mathcal{P} = \langle\, N, \sqsubseteq \,\rangle$ if there is a standard labelling $l(\cdot)$ of $\mathcal{P}$ such that $\{\, l(s) \,:\, s \in S \,\} = \{\, |\, N \,| - |\, S \,| + 1, \ldots, |\, N \,| \,\}$. (such that $\{\, l(s) \,:\, s \in S \,\} = \{\, 1, \ldots, |\, S \,| \,\}$).

Theorem 4. *Let* $P = \langle N, R \rangle$ *be a partial order let* $1 \leq s \leq |\, N \,|$ *and let* M *be the set constaining all maximal sets of* $\mathcal{P}$ *that have a Cardinality of* s*, then*

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

Proof. By induction on s and application of Theorem 3.

Theorem 4 remains true if minimal is substituted for maximal. The algorithm called *frequency′* (depicted in Figure 2) uses Theorem 4 for frequency computation. The parameter *size* is the *size-parameter* of the algorithm. It corresponds to s in Theorem 4.

```
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. 2. Improved algorithm for computing the frequency of partial orders.

5 Experimental Results

Let $K_{m,n} = \langle \{1, \ldots, m+n\}, \sqsubseteq \rangle$, where $i \sqsubset j \iff i \leq m < j$. These orders are difficult to count for *frequency* and *frequency*$'$. "Tree shaped orders" are among the easiest connected orders count. 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 $\mathrm{f}(B_n^+) = \mathrm{f}(B_n^-)$.

We implemented *frequency* and *frequency*$'$ in `Prolog`. They improved significantly over the MAC-based algorithms. MAC-3_d our best MAC-based algorithm required more than 6 hours for computing $\mathrm{f}(K_{8,8})$. An implementation with OPL did not terminate after many hours and required an intermediate memory size of more than 60MB. However, *frequency*$'$ required 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 *frequency*$'$. All results were obtained with a 1000 MHz DELL Latitude. The results for B_n^- with a size-parameter of 1 demonstrate the advantage of using Theorem 1. Each time a node is removed from B_n^- this results in two B_{n-1}^-, for $n > 0$. The results for the $K_{n,n}$ and B_n^+ demonstrate the advantage of Theorem 4 because as the size-parameter increases less and less time is required.

The differences in time for computing the frequencies of B_n^+ and B_n^- demonstrates that *frequency*$'$ is not clever at exploiting structural properties of partial orders. It should be possible to improve the algorithm by allowing it to also remove maximal sets.

Problem	Size	Frequency	Time in Seconds
B_3^+	1–3	stack overflow	—
B_3^+	4	21964800	3.45
B_3^-	1	21964800	0.00
B_6^-	1	$10^{163.61}$	0.21
B_8^-	1	$10^{9568.46}$	3.96
$K_{8,8}$	1	stack overflow	—
$K_{8,8}$	2	1625702400	1.96
$K_{8,8}$	8	1625702400	0.01
$K_{10,10}$	10	13168189440000	0.06
$K_{16,16}$	16	437763136697395052544000000	9.30

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

6 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 $\mathcal{P}$. For moderately sized problems the techniques presented in this paper significantly reduce the total solution time.

References

1. 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.
2. 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.
3. 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.
4. M.P. Schellekens. Compositional average time analysis *toward a calculus for software timing*. Technical report, Centre for Efficiency Orientated Languages, 2004. In Preparation.
5. R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Publishing Company, 1996.
6. M.R.C. van Dongen. Saving support-checks does not always save time. *Artificial Intelligence Review*, 2004. Accepted for publication.
7. P. Van Hentenryck. *The OPL Programming Language*. MIT Press, 1999.