Constraint Programming Letters 2 (2008) 55–77 Submitted 3/2007; Published 1/2008

The Expected Value and the Variance of the Checks Required by Revision Algorithms

M.R.C. van Dongen DONGEN@CS.UCC.IE
A.B. Dieker T.DIEKER@PROBA.UCC.IE
University College Cork
Western Road
Cork
Ireland

A. Sapozhnikov ARTEM.SAPOZHNIKOV@CWI.NL
Centrum voor Wiskunde en Informatica
Kruislaan 413
1098SJ Amsterdam
The Netherlands

Editor: M.R.C. van Dongen

Abstract

This paper presents the analysis of three revision algorithms for 2-variable Constraint Satisfaction Problems (CSPs). The revision algorithms under consideration are called $\mathcal{L}$, $\mathcal{L}'$, and $\mathcal{L}''$. For 2-variable CSPs $\mathcal{L}$ models an optimal arc consistency algorithm which exploits multi-directionality. However, $\mathcal{L}'$ and $\mathcal{L}''$ do not exploit multi-directionality. For 2-variable CSPs $\mathcal{L}'$ is equivalent to the arc consistency algorithm AC-3. The expected number and the variance of the checks are presented for $\mathcal{L}$, $\mathcal{L}'$, and $\mathcal{L}''$. Writing $A \prec B$ if the expected number of checks for A is less than for B, we have $\mathcal{L} \prec \mathcal{L}' \prec \mathcal{L}''$. The results are parametrised over the probability p that a random check succeeds and probability $q = 1-p$ that it fails. It is proved that the difference between the expected number of checks of any two algorithms is bounded by $\min(b, 1 + \lceil \ln(a)/\ln(1/q) \rceil)/p$. Using a variance analysis, it is proved that, as the domain sizes a and b become large, the number of checks which are required by the revision algorithms is sharply concentrated about their expected number of checks. Finally, our analysis allows us to find an upper bound of $2ed/p + (2e-n)d(d-1)/2p$ for the expected time complexity of AC-3, where e is the number of constraints, n is the umber of variables, and d is the maximum domain size. These results are novel and encouraging. First they provide the first non-trivial upper bound on the expected time complexity of AC-3. Second, they demonstrate that on average there is a small margin separating $\mathcal{L}$, $\mathcal{L}'$, and $\mathcal{L}''$. Finally, they present the first results about the variance of the checks required by revision algorithms.

Keywords: Filtering Algorithms, Arc Consistency, Time Complexity, Average Time Complexity

1. Introduction

Arc consistency algorithms (Mackworth, 1977) are a major component of many industrial and academic Constraint Satisfaction Problem (CSP) solvers. These algorithms ensure that each value in the domain of each variable is supported by some value in the domain of each variable by which it is constrained. Sitting at the heart of a CSP solver, arc consistency algorithms consume a large portion

©2008 M.R.C. van Dongen, A.B. Dieker, and A. Sapozhnikov.

of the time which is required to solve the input CSP. For these reasons arc consistency algorithms and their time complexity always has been a heavily researched area.

In this paper we restrict our attention to binary CSPs. For historical reasons the *theoretical* time complexity of arc consistency algorithms is usually measured in the number of *consistency checks* which are required by the algorithms. Each check consists of a "lookup operation" to find if a pair of values are allowed by a constraint. The theoretical worst-case time complexity of binary arc consistency algorithms is $\mathcal{O}\left(ed^2\right)$ (Mohr and Henderson, 1986), where e is the number of constraints and d is the maximum domain size. This is optimal in the worst-case. From now on we write *(non-)optimal* for *worst-case (non-)optimal*.

Some researchers have noticed discrepancies between the optimal time complexity and the *average observed* running time of arc consistency algorithms. Wallace (1993) was the first to point out that the optimal algorithm AC-4 (Mohr and Henderson, 1986) has a worse *average observed* running time than the non-optimal algorithm AC-3 (Mackworth, 1977). Similarly, Van Dongen (2004) experimentally compared the optimal algorithm AC-2001 (Bessière and Régin, 2001) and several non-optimal AC-3 (Mackworth, 1977) style algorithms. His results also indicate that the non-optimal algorithms have a better *average* observed running time.

Since the introduction of the optimal algorithm AC-4 (Mohr and Henderson, 1986) there has been a thrust of work on the design of optimal arc consistency algorithms which have a better theoretical and observed *average* running time than the bound $\mathcal{O}\left(ed^2\right)$ (Bessière and Cordier, 1993; Bessière and Régin, 1994; Bessière et al., 1995; Bessière and Régin, 2001; Zhang and Yap, 2001; Lecoutre et al., 2003; Lecoutre and Hemery, 2007). The main mechanism for improvements are avoiding repeated checks, postponing checks until they are really needed, and exploiting that checks are bi-directional or multi-directional (Bessière et al., 1995; Lecoutre et al., 2003). Each improvement results in a better theoretical and observed *average* running time.

These findings suggests that the theoretical and observed average running time is important. To date the theoretical average time complexity of any arc consistency algorithm remains unknown. However, more is known for the case of 2-variable CSPs. Here a *2-variable* CSP is a CSP having two variables and a single binary constraint. For 2-variable CSPs the theoretical average time complexity of two optimal *revision* (Mackworth, 1977) algorithms is studied by Van Dongen (2002). Here a *revision algorithm* is an algorithm which enforces arc consistency in both domains of a 2-variable CSP. The *tightness* is the percentage of disallowed tuples in the Cartesian product of the domains. Lecoutre and Hemery (2007) notice that AC-3 equipped with residues requires two checks (asymptotically) if the tightness is 0.5.

Studying the average time complexity of 2-variable CSPs is important since an improved bound which is obtained for the average complexity for the 2-variable CSP may be used to obtain improved upper bound for the average time complexity of AC-3. This is demonstrated in Section 6. AC-3 never spends fewer checks than the optimal algorithm AC-2001/AC-3.1. Therefore, the bounds from Section 6 are also upper bounds for the average time complexity of AC-2001/AC-3.1.

This paper proposes a new approach to the problem of determining the average time complexity of arc consistency algorithms for 2-variable CSPs. We compute *probability generating functions* (PGFs), which enable us to compute the expected number of checks required by three revision algorithms which are called $\mathcal{L}$, $\mathcal{L}'$, and $\mathcal{L}''$. $\mathcal{L}$ models an arc consistency algorithm which exploits bi-directionality. $\mathcal{L}'$ and $\mathcal{L}''$ do not exploit bi-directionality. For two variables $\mathcal{L}$ is equivalent to AC-3.3 (Lecoutre et al., 2003) and $\mathcal{L}'$ is equivalent to AC-3 (Mackworth, 1977). $\mathcal{L}''$ is the least efficient of all algorithms. The main reason for studying it is that it allows us to compare $\mathcal{L}'$ and

$\mathcal{L}$. $\mathcal{L}$ is also studied by Van Dongen (2002). The PGFs are parametrised: they assume Bernoulli distribution with parameter p that a given pair of values is allowed by a constraint and probability $q = 1 - p$ that the pair is disallowed. This assumption corresponds to random CSPs which are generated according to Model A or D (Gent et al., 2001). This paper is a continuation of the work presented in (Van Dongen, 2002) in the following sense. First, we generalise the implicit assumption from Van Dongen (2002) that $p = 1/2$. Second, we present generating functions and variances of the checks. Third, we analyse the algorithms $\mathcal{L}'$ and $\mathcal{L}''$. Here $\mathcal{L}'$ corresponds to two consecutive applications of the *revise* algorithm (Mackworth, 1977). Finally, we compare $\mathcal{L}$, $\mathcal{L}'$, and $\mathcal{L}''$.

Using a generating function approach in the analysis of consistency algorithms is new and interesting. For example, given the generating functions of the algorithms we can obtain not only their expected time complexities but also any higher moment. In our case this allows us, for the first time, to obtain the variance of the three algorithms under consideration. In addition we prove that the variances are low. Intuitively this demonstrate that all algorithms are robust, in the sense that the number of checks which is required for a random constraint does not deviate "much" from the expected number of checks.

We now summarise our results on average time complexity for the revision algorithms. Let a and b be the domain sizes of the variables. Furthermore, let $N_{\mathcal{A}}$ be the number of checks which are required by revision algorithm $\mathcal{A}$ for a random $a \times b$ constraint. Finally, let $\mathbb{E}(N_{\mathcal{A}})$ be the expected value of $N_{\mathcal{A}}$. Writing $\mathcal{A} \prec \mathcal{B}$ if $\mathbb{E}(N_{\mathcal{A}}) < \mathbb{E}(N_{\mathcal{B}})$ we have $N_{\mathcal{L}} \prec N_{\mathcal{L}'} \prec N_{\mathcal{L}''}$. We prove that $\mathbb{E}(N_{\mathcal{L}''}) = [\, a(1-q^b)+b(1-q^a)\,]/p$ and that $\mathbb{E}(N_{\mathcal{L}}) = [\, a(1-q^b-bpq^b)-bq^a+\sum_{\ell=0}^{b-1}(1-pq^\ell)^a\,]/p$. The expression for $\mathbb{E}(N_{\mathcal{L}'})$ is not presented in this section. It is proved that $\mathbb{E}(N_{\mathcal{L}''}) - \mathbb{E}(N_{\mathcal{L}}) = \sum_{\ell=0}^{b-1}[\,1-(1-pq^\ell)^a\,]/p \leq U_{a,b} = \min(b, 1 + \lceil \ln(a)/\ln(1/q)\,\rceil)/p$. Since $\mathcal{L} \prec \mathcal{L}' \prec \mathcal{L}''$ it follows that $0 \leq \mathbb{E}(N_{\mathcal{L}'}) - \mathbb{E}(N_{\mathcal{L}}) \leq U_{a,b}$. It is recalled that $\mathcal{L}$ is the best algorithm and that it exploits multi-directionality. The worst algorithm, $\mathcal{L}''$, carries out two independent revisions. In the second revision it forgets all the checks from the first revision. Our findings suggest that for the two-variable random CSP the difference between the checks which are required by $\mathcal{L}''$ and $\mathcal{L}$ is small on the average.

Variances are not presented in this section. However, we are able to demonstrate that, as the domain sizes a and b become large, the number of checks required by the algorithms is sharply concentrated about their expected values. Finally, our analysis allows us to find the upper bound $2ed/p + (2e - n)d(d-1)/2p$ for the expected time complexity of AC-3 for the general n-variable CSP.

Most derivations for the PGFs are presented in the main text. Much of the remaining derivations and proofs are presented in the appendix. The remainder of the paper is as follows. Section 2 presents a description of the algorithms. Section 3 presents the derivation of two probability generating functions. The first PGF is for the checks required to locate the first 1 in a sequence of zeros and ones. The second PGF is for the checks required by $\mathcal{L}$. This is followed by Section 4, which presents the expected number of checks and the variance of the checks of the three revision algorithms. Section 5 compares their expected running time and their variance. The upper bound for the expected time complexity of AC-3 is provided in Section 6. Conclusions are presented in Section 7.

2. Description of the Algorithms

This section describes the arc consistency algorithm AC-3 and the revision algorithms $\mathcal{L}$, $\mathcal{L}'$, and $\mathcal{L}''$. To make the paper accessible to a wider audience, a constraint is a Boolean matrix. This allows us to view the problem of making a 2-variable CSP arc consistent as that of finding the non-zero rows and columns in a given matrix. Before presenting the algorithms, we remind the reader of some basic definitions about CSPs and arc consistency.

Formally, a binary CSP is a tuple $(\, X, D, C \,)$, where X is a set of n variables, $D(x)$ is the domain of variable x, and C is a set of $2e$ *directed constraints* between pairs of variables. If $con \in C$ is the constraint between two variables v and w then C also contains the constraint con^T between w and v, where $\cdot^T$ denotes transposition. $D(v)$ is *arc consistent* with respect to w if $1 \in \{\, con[\, i, j \,] : j \in D(w) \,\}$, for each $i \in D(v)$. $D(v)$ is *arc consistent* if it is arc consistent with respect to all variables by which v is constrained. A CSP is *arc consistent* if its domains are arc consistent.

Pseudo-code for the revision algorithms is depicted in Figures 3–5. They all have the same parameters: the domain A of a variable, the domain B of another variable, and the constraint con between the variables. All algorithms remove $i \in A$ from A if and only if $con[\, i, j \,] = 0$ for all $j \in B$. Likewise they remove $j \in B$ from B if and only if $con[\, i, j \,] = 0$ for all $i \in A$.

All revision algorithms use Algorithm $initialise$, which is depicted in Figure 1. All algorithms also use Algorithm $revise$, which is depicted in Figure 2. This algorithm uses the constants $failure$, $success$, and $unchecked$, which are all different. The time complexity of arc consistency algorithms is measured by the number of consistency checks which are required by the algorithms. This is exactly the number of times the algorithms index a constraint or its transpose. Constraints are indexed only in $revise$. All checks for $\mathcal{L}$ are cached in the 2-dimensional array $checked$. $\mathcal{L}'$ and $\mathcal{L}''$ also cache their checks but they do not exploit this.

Algorithm $\mathcal{L}$ starts by computing its *row support*, which is the set consisting of the indices of the non-zero rows, i.e. all $i \in A$ such that $con[\, i, j \,] = 1$ for some $j \in B$. $\mathcal{L}$ continues by computing its *column support*, which is the set consisting of the indices of the non-zero columns, i.e. all $j \in B$ such that $con[\, i, j \,] = 1$ for some $i \in A$. Next $\mathcal{L}$ terminates. The checks required for computing the row (column) support are called the *row (column) support checks*.

The algorithms $\mathcal{L}'$ and $\mathcal{L}''$ are similar to $\mathcal{L}$ but they sometimes repeat their checks. For 2-variable CSPs $\mathcal{L}$ carries out the same amount of work as is required by the optimal arc consistency algorithm AC-3.3 (Lecoutre et al., 2003). In terms of the number of consistency checks it is the most efficient of the three revision algorithms. $\mathcal{L}'$ always spends more checks than $\mathcal{L}$. The checks which $\mathcal{L}'$ may repeat are checks of the form $con[\, i, j \,]$, where i is a non-empty row and $con[\, i', j \,] = 0$, for all $i' \in A$ such that $i' < i$. For example, it always repeats the check $con[\, \min(A), \min(B) \,]$ if the row $\min(A)$ is non-zero. For 2-variable CSPs the work which is required by $\mathcal{L}'$ is the same as that which is required by the algorithm AC-3 (Mackworth, 1977). The algorithm $\mathcal{L}''$ is less efficient than $\mathcal{L}'$ because it always carries out the same checks as $\mathcal{L}'$ but sometimes carries out more checks. For example, it may repeat checks of the form $con[\, i, j \,]$, where $con[\, i', j \,] = 0$ for all $i' \in B$, which is never done by $\mathcal{L}'$. It follows that qualitatively we have $\mathcal{L} \prec \mathcal{L}' \prec \mathcal{L}''$.

Pseudo-code for Algorithm AC-3 (Mackworth, 1977) is depicted in Figure 6. Its single argument is the *directed constraint graph*, G. This is a digraph having $2e$ arcs. For each constraint between a pair of variables v and w it has the arcs $(\, v, w \,)$ and $(\, w, v \,)$. The algorithm starts by initialising the set Q, which is known as the *queue* in the literature. After this initialisation the queue consists

```
procedure initialise( set of int rsup, set of int csup,
                      set of int A, set of int B,
                      int checked[ , ] ): begin
   rsup := ∅;
   csup := ∅;
   forall i ∈ A do
      forall j ∈ B do
         checked[ i, j ] := unchecked;
end;
```

Figure 1: Pseudo-code for Algorithm *initialise*.

```
procedure revise( set of int rsup, set of int csup,
                  set of int A, set of int B,
                  int checked[ , ], int con[ , ] ): begin
   forall i ∈ A do
      forall j ∈ B do begin
         if i ∈ rsup then break;
         if checked[ i, j ] = unchecked then begin
            if con[ i, j ] = 0 then
               checked[ i, j ] := failure;
            else begin
               found := true;
               checked[ i, j ] := success;
               rsup := rsup ∪ { i };
               csup := csup ∪ { j };
            end;
         end;
      end;
end;
```

Figure 2: Pseudo-code for Algorithm *revise*.

```
procedure L( set of int A, set of int B, int con[ , ] ): begin
   initialise(rsup, csup, A, B, checked);
   revise(rsup, csup, A, B, checked, con);
   A := rsup;
   revise(csup, rsup, B, A, checked^T, con^T);
   B := csup;
end;
```

Figure 3: Pseudo-code for Algorithm $\mathcal{L}$.

```
procedure L'( set of int A, set of int B, int con[ , ] ): begin
   initialise(rsup, csup, A, B, checked);
   revise(rsup, csup, A, B, checked, con);
   A := rsup;
   initialise(rsup, csup, A, B, checked');
   revise(csup, rsup, B, A, checked'^T, con^T);
   B := csup;
end;
```

Figure 4: Pseudo-code for Algorithm $\mathcal{L}'$.

```
procedure L''( set of int A, set of int B, int con[ , ] ): begin
   initialise(rsup, csup, A, B, checked);
   initialise(rsup', csup', A, B, checked');
   revise(rsup, csup, A, B, checked, con);
   revise(csup', rsup', B, A, checked'^T, con^T);
   A := rsup;
   B := csup;
end;
```

Figure 5: Pseudo-code for Algorithm $\mathcal{L}''$.

```
function AC-3( directed graph G ): begin
   Q := G;
   while Q ≠ ∅ do begin
      select and remove any arc ( v, w ) from Q;
      initialise(rsup, csup, D(v), D(w), checked);
      con := the (directed) constraint between v and w;
      revise(rsup, csup, D(v), D(w), checked, con)
      if rsup = ∅ then
         return false;
      else if D(v) ≠ rsup then begin
         D(v) := rsup;
         Q := Q ∪ { ( w', v ) ∈ G : w ≠ w' };
      end;
   end;
   return true;
end;
```

Figure 6: Pseudo-code for Algorithm AC-3.

of $2e$ arcs — one for each directed constraint. The algorithm continues by selecting and removing any arc, (v, w), from the queue and revising the domain of v against that of w. If the domain of v becomes empty then this proves that the arc consistent equivalent of the input CSP has empty domains and the algorithm will immediately terminate. Otherwise, if one or more values have been removed from $D(v)$ then the algorithm propagates the consequences of this change by adding arcs to the queue. It is noted that it follows that if the domain of v is revised for the i-th time against the domain of w then at least $i-1$ values have been removed from the (input) domain of w.

It should be noted that all algorithms are presented so as to facilitate ease of comparison in terms of the number of checks which they require. They are not intended to be efficient.

3. The Probability Generating Functions of $\xi^{(\ell)}$ and $N_{\mathcal{L}}$

This section presents the main derivation of two *probability generating functions* (PGFs). The first one is the PGF of the number of checks which are required to locate the first 1 in a sequence of zeros and ones. The second is the PGF of the number of checks which are required by Algorithm $\mathcal{L}$. Basic facts about PGFs are presented in Section 3.1. Section 3.2 defines the random variable $\xi^{(\ell)}$ which measures the required effort for locating the first one in a random sequence consisting of ℓ zeros and ones. Section 3.2 also computes the probability generating function of $\xi^{(\ell)}$, its first two moments, and its variance. Section 3.3 proceeds by studying the patterns arising in the checks that are carried out by $\mathcal{L}$ and by defining the random variable $N_{\mathcal{L}}$ which measures the number of checks which are required by $\mathcal{L}$ for a random $a \times b$ constraint. Section 3.4 presents the derivation of the probability generating function of $N_{\mathcal{L}}$.

3.1 Basic Facts

This section presents notation and some facts about *probability generating functions*.

Let X be a random variable. We shall write $\mathbb{P}(X = k)$ for the probability that X equals k, $\mathbb{E}(X)$ for the expected value of X, and $\mathbb{V}\mathrm{ar}(X) = \mathbb{E}\left(X^2\right) - (\mathbb{E}(X))^2$ for the variance of X. The *probability generating function* (PGF) of a random nonnegative integer variable X is defined as $\sum_{n=0}^{\infty} \mathbb{P}(X = n)\, z^n$. We shall write $\mathbb{E}z^X$ for the probability generating function of X. The generating function of X provides all information about the expected value of X and any higher moment. For example, the following two equivalences describe how to obtain the expected value of X and that of X^2 from the PGF of X (Graham et al., 1989, Equations 8.28 and 8.29).

$$\mathbb{E}(X) = \frac{d}{dz}\mathbb{E}z^X\bigg|_{z=1}, \tag{1}$$

$$\mathbb{E}\left(X^2\right) = \frac{d}{dz}\mathbb{E}z^X\bigg|_{z=1} + \frac{d^2}{dz^2}\mathbb{E}z^X\bigg|_{z=1}. \tag{2}$$

More information about generating functions may be found in (Graham et al., 1989; Sedgewick and Flajolet, 1996; Wilf, 1994).

3.2 The Distribution of $\xi^{(\ell)}$

Let ℓ be a nonnegative integer. Consider a random sequence consisting of ℓ zeros and ones. We wish to measure the *average* effort (accumulated cost) which is required to locate the first one in the sequence (or deciding that the row consists of zeros only). Here locating the first one in the

sequence means looking from start to end in the sequence, checking each member at most once, and stopping as soon as we have located the first one. Each time we carry out a check, our accumulated cost increases by one. We start with an accumulated costs of zero.

Clearly, the average number of checks depends on the distribution of the zeros and ones in the sequence. In the following, we shall assume a Bernoulli distribution for the members in the sequence. We shall assume that the distribution of each member is independent and that a member equals 1 with probability p and equals 0 with probability $q = 1 - p$.

We define $\xi^{(\ell)}$ the number of checks which are required to find the first 1 in a random sequence consisting of ℓ zeros and ones. Using $\xi^{(\ell)}$ the average number of checks may now be expressed as the expected value of $\xi^{(\ell)}$. Notice that by definition we have $\xi^{(1)} = 1$, since the number of required checks for any sequence of length one equals 1. Likewise, we have $\xi^{(0)} = 0$.

It immediately follows that the distribution of $\xi^{(\ell)}$ is given by

$$\mathbb{P}\left(\xi^{(\ell)} = m\right) = \begin{cases} q^{m-1}p & \text{if } 1 \le m < \ell\,, \\ q^{\ell-1} & \text{if } 1 \le m = \ell\,, \\ 1 & \text{if } 0 = m = \ell\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

With this we readily calculate the PGF of the distribution of $\xi^{(\ell)}$: for $z \in [0:1)$, we have

$$\mathbb{E}z^{\xi^{(\ell)}} = \sum_{m=0}^{\ell} \mathbb{P}\left(\xi^{(\ell)} = m\right) z^m = q^{\ell-1}z^\ell + \sum_{m=1}^{\ell-1} q^{m-1}pz^m = \frac{1-z}{1-qz}\,(qz)^\ell + \frac{pz}{1-qz}\,.$$

Using Equations (1) and (2) we readily compute the first two moments and the variance of $\xi^{(\ell)}$.

$$\mathbb{E}\left(\xi^{(\ell)}\right) = \frac{1-q^\ell}{p}\,, \tag{3}$$

$$\mathbb{E}\left(\left(\xi^{(\ell)}\right)^2\right) = \frac{2-p+q^\ell(p-2-2p\ell)}{p^2}\,, \tag{4}$$

$$\mathbb{V}\mathrm{ar}\left(\xi^{(\ell)}\right) = \frac{q+q^\ell(p-2p\ell-q^\ell)}{p^2}\,. \tag{5}$$

3.3 The General Pattern

Before starting with the derivation of the PGF of $N_{\mathcal{L}}$, it is instructive to study the general pattern of the checks which are carried out by $\mathcal{L}$. The computation of the row support gives rise to a unique *pattern* which is a matrix consisting of zeros, ones, asterisks ($*$) and minuses ($-$). The zeros and the ones are all entries of the matrix which are revealed by the computation of the row support. The asterisks are all unchecked locations that are possible candidates for the column support checks. The remaining unchecked locations contain a minus. They are never checked by $\mathcal{L}$.

Figure 7 depicts a possible pattern for $a = b = 5$. The number of ones and zeros in the pattern is exactly equal to the number of row support checks required by $\mathcal{L}$.

The number of row support checks of the pattern does not change if we permute the rows of our pattern. Sorting the rows in reverse lexicographic order we obtain a pattern which we call the *standard form* of the pattern. Figure 8 depicts the standard form of the pattern which is depicted in Figure 7. The expected number of checks of any pattern and its standard form are the same.

$$\begin{bmatrix} 1 & * & * & - & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 7: Pattern of a 5×5 matrix.

$$\begin{bmatrix} 1 & * & * & - & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 8: Standard form of pattern.

The standard form of a pattern makes it very easy to read off the expected number of column support checks of that pattern. To see how this works, let m_j denote the number of non-zero rows having their first 1 in column j. The number m_j is called the *multiplicity* of column j. For our example we have $m_1 = 1$, $m_4 = 2$ and $m_2 = m_3 = m_5 = 0$. The first stage of the algorithm consists of the computation of the row support checks. After this stage we know the number of non-zero rows, c, and the multiplicity, m_j, of column j. We also know the number k_j of unchecked elements in column j for which $m_j = 0$. These unchecked elements are the asterisks in Figures 7 and 8. We define $k_j = 0$ if $m_j \neq 0$. Using this notation, the expected number of column support checks is equal to $\sum_{j=1}^{b} \mathbb{E}\left(\xi^{(k_j)}\right)$, where $\mathbb{E}\left(\xi^{(k_j)}\right)$ is the expected number of checks which are required to find the first one in a random length-k_j sequence of zeros and ones. In our example, the expected number of checks required for the matrices having the pattern from Figure 8 is equal to $2 \times \mathbb{E}\left(\xi^{(0)}\right) + 2 \times \mathbb{E}\left(\xi^{(1)}\right) + \mathbb{E}\left(\xi^{(3)}\right)$.

Given the number of non-zero rows, c, the definition of m_j gives rise to an order-c *multiplicity sequence*, which is a sequence $\langle\, m_1, \ldots, m_b \,\rangle \in \mathbb{N}^b$ such that $\sum_{j=1}^{b} m_j = c$. Each standard form has a unique multiplicity sequence, which allows us to compute the number of forms having the same standard from. Fixing an order-c multiplicity sequence $\langle\, m_1, \ldots, m_b \,\rangle$, the number of patterns having the same multiplicity sequence is equal to the number of partitions of a set of a elements into $b+1$ subsets of size $a-c, m_1, m_2, \ldots, m_b$. There are exactly $\binom{a}{c} \times \binom{c}{m_1,\ldots,m_b}$ such partitions, where $\binom{c}{m_1,\ldots,m_b} = \frac{c!}{(m_1!)\times\ldots\times(m_b!)}$ is the *multinomial coefficient* of c and $m_1, \ldots, m_b$. This facilitates the computation of our PGF. Notice that in the definition of the multinomial coefficient $\binom{c}{m_1,\ldots,m_b}$ it is assumed, as it is in our case, that $c = \sum_{j=1}^{b} m_j$.

3.4 The Derivation of the PGF of $N_{\mathcal{L}}$

Having studied the general pattern which is revealed by the computation of the row support, it is now relatively easy to derive the PGF of $N_{\mathcal{L}}$. Throughout we write $\sum_{m_1+\ldots+m_b=c} f(m_1, \ldots, m_b)$ for the sum of $f(m_1, \ldots, m_b)$ over all possible nonnegative integers $m_1, \ldots m_b$ satisfying the constraint $m_1 + \ldots + m_b = c$.

Our general PGF is of the form $\mathbb{E}z^{N_{\mathcal{L}}} = \sum_{n\in\mathbb{N}} \mathbb{P}\left(N_{\mathcal{L}} = n\right) z^n$, but we shall compute it by summing over multiplicity sequences. This gives rise to an expression of the form

$$\mathbb{E}z^{N_{\mathcal{L}}} = \sum_{c=0}^{a} \sum_{m_1+\ldots+m_b=c} p(\langle\, m_1, \ldots, m_b \,\rangle) \times C_{\langle\, m_1,\ldots,m_b \,\rangle}(z) \;, \tag{6}$$

where $p(m)$ is the probability that a random matrix has multiplicity sequence m and $C_m(z)$ is the PGF of the row support checks which are spent using a random $a \times b$ matrix having multiplicity sequence m. Here $C_m(z)$ naturally can be written $L_m(z) \times R_m(z)$, where $L_m(z)$ is the PGF for

the row support checks and $R_m(z)$ is the PGF for the column support checks. We note that

$$p(\langle m_1, \ldots, m_b \rangle) = \binom{a}{c} \binom{c}{m_1, \ldots, m_b} p^c q^{b\times(a-c)-c+\sum_{j=1}^{b} jm_j} .$$

Select any number, c, of rows, any order-c multiplicity sequence, m, and any matrix M having c non-zero rows and having multiplicity sequence m. Then $\mathcal{L}$ requires exactly $b\times(a-c)+\sum_{j=1}^{b} jm_j$ row support checks for M. Therefore, $L_m(z)$ is given by

$$L_m(z) = z^{b\times(a-c)} \prod_{j=1}^{b} z^{jm_j} .$$

The expected number of column support checks which are required for the j-th column of M is equal to the expected value of $\xi^{\left(\sum_{k=1}^{j} m_k 1_{\{m_j=0\}}\right)}$. Here $1_{\{P\}}$ is the *indicator function* of the predicate P: it is 1 if P is true and 0 otherwise. The PGF, $R_m(z)$, of the column support checks is given by

$$R_m(z) = \prod_{j=1}^{b} \mathbb{E} z^{\xi^{\left(\sum_{k=1}^{j} m_k 1_{\{m_j=0\}}\right)}} .$$

Having computed $L_m(z)$ and $R_m(z)$ it follows that

$$\mathbb{E} z^{N_{\mathcal{L}}} = \sum_{c=0}^{a} \binom{a}{c} (qz)^{b(a-c)} (p/q)^c f_z(b,c) , \tag{7}$$

where

$$f_z(b,c) = \sum_{m_1+\ldots+m_b=c} \binom{c}{m_1, \ldots, m_b} \prod_{j=1}^{b} \mathbb{E} z^{\xi^{\left(\sum_{k=1}^{j} m_k 1_{\{m_j=0\}}\right)}} \prod_{j=1}^{b} (qz)^{jm_j} .$$

We conclude this section by showing how $f_z(b,c)$ may be computed recursively. First, we have

$$f_z(b,c) = \sum_{m_1+\ldots+m_b=c} \binom{c}{m_1, \ldots, m_b} \prod_{j=1}^{b} \left(\mathbb{E} z^{\xi^{\left(\sum_{k=1}^{j} m_k\right)}} 1_{\{m_j=0\}} + (qz)^{jm_j} 1_{\{m_j\neq 0\}} \right) . \tag{8}$$

By separately considering $m_b = 0$ and $m_b > 0$, this may be computed efficiently as follows

$$f_z(b,c) = \mathbb{E} z^{\xi^{(c)}} f_z(b-1,c) + \sum_{k=1}^{c} \binom{c}{k} (qz)^{bk} f_z(b-1, c-k) . \tag{9}$$

4. The Expected Value and the Variance of the Algorithms

This section presents the expected value and the variance of the checks which are required by $\mathcal{L}$, $\mathcal{L}'$, and $\mathcal{L}''$. It is recalled that the random variable $N_{\mathcal{A}}$ counts the number of checks which are required by $\mathcal{A}$ so as to find the non-zero rows and columns in a random $a \times b$ matrix. The expected number of checks of $\mathcal{A}$ may then be expressed as $\mathbb{E}(N_{\mathcal{A}})$ and the variance of the running time as $\mathbb{V}\mathrm{ar}(N_{\mathcal{A}})$.

4.1 Algorithm $\mathcal{L}$

This section presents the complexity results for $\mathcal{L}$. We start with the expected number of checks. We define $F(0,b)=0$ for $b\geq 1$ and for $a,b\geq 1$, we set

$$F(a,b)=\frac{a(1-q^b-bpq^b)-bq^a}{p}+\frac{1}{p}\sum_{\ell=0}^{b-1}(1-pq^\ell)^a\,. \tag{10}$$

The following propositions are our main results for $\mathcal{L}$. Proof may be found in Appendix A.

Proposition 1 (Expected Value of $N_{\mathcal{L}}$) *For $a,b\geq 1$, we have*

$$\mathbb{E}\left(N_{\mathcal{L}}\right)=abq^b+F(a,b)=a\frac{1-q^b}{p}+b\frac{1-q^a}{p}+\frac{1}{p}\sum_{\ell=0}^{b-1}\Big((1-pq^\ell)^a-1\Big)\,.$$

Proposition 2 (Variance of $N_{\mathcal{L}}$) *For $a,b\geq 1$, we have*

$$\mathbb{V}\mathrm{ar}\left(N_{\mathcal{L}}\right)=\mathbb{E}\left(N_{\mathcal{L}}^2\right)-\Big(abq^b+F(a,b)\Big)^2\,,$$

with

$$\mathbb{E}\left(N_{\mathcal{L}}^2\right)=a(a-1)\left[b^2q^{2b}+p^2\right]+ab^2q^b+F(a,b)+2abq^bF(a-1,b)+\sum_{\ell=2}^{b}h(\ell)\,,$$

where for $\ell\geq 2$,

$$\begin{aligned}h(\ell)=\;&2a\ell pq^{\ell-1}F(a-1,\ell-1)+a(a-1)\ell^2p^2q^{2\ell-2}+a\ell(\ell-1)pq^{\ell-1}\\&+2a\frac{1-q^{\ell-1}-(\ell-1)pq^{\ell-1}}{p^2}(1-pq^{\ell-1})^{a-1}+\frac{2}{p^2}\sum_{k=0}^{\ell-2}(1-pq^{\ell-1}-pq^k)^a\\&-2(\ell-1)\frac{(q-pq^{\ell-1})^a}{p^2}-\frac{2q^aF(a,\ell-1)}{p}+\frac{2q(1-pq^{\ell-1})^a}{p^2}\\&-\frac{2q^{a+1}}{p^2}-\frac{2a(1-q^{\ell-1})q^a}{p}\,.\end{aligned}$$

4.2 Algorithm $\mathcal{L}'$

This section presents the complexity results for $\mathcal{L}'$. As with $\mathcal{L}$ we express our results in terms of a random variable, $N_{\mathcal{L}'}$, which counts the number of checks which are required by Algorithm $\mathcal{L}'$ so as to find the non-zero rows and columns in a random $a\times b$ matrix. The key observation is that $N_{\mathcal{L}'}=N_{\mathcal{L}'}^{(1)}+N_{\mathcal{L}'}^{(2)}$, with

$$N_{\mathcal{L}'}^{(1)}=\gamma_1^{(b)}+\ldots+\gamma_a^{(b)}\,,\quad N_{\mathcal{L}'}^{(2)}=\eta_1^{(a)}+\ldots+\eta_b^{(a)}\,, \tag{11}$$

where the γ correspond to the row support checks and the η correspond to the column support checks. All the $\gamma^{(b)}$ are mutually independent and identically distributed as $\xi^{(b)}$, and the $\eta^{(a)}$ are identically distributed. For $1\leq k\leq a$ we have

$$\mathbb{P}\left(\eta_j^{(a)}=k\right)=\binom{a}{k}q^{b(a-k)}(q-q^b)^k+\sum_{c=0}^{a-k}\binom{c+k-1}{c}q^{cb}(q-q^b)^{k-1}p\,. \tag{12}$$

It is noted that it is possible for $\eta_j^{(a)}$ to be zero. The first term corresponds to column j containing only zeros. The key observation is that $\mathcal{L}'$ does not carry out checks in rows with only zeros. Therefore, if there is no zero in column j, we must have $a-k$ zero-rows (the probability of such a row is q^b) and k rows with at least one 1 but not in column j (the probability of such a row is $q-q^b$). The zero-rows can be arbitrarily placed, whence the binomial term. Next consider the case where there is a 1 in the j-th column. The sum of Equation (12) is taken over the number $c \leq a-k$ zero-rows before the first 1 in this column. We remark that this sum can be rewritten in terms of hypergeometric functions, but we prefer a finite-sum representation.

The following two propositions are our main results for $\mathcal{L}'$. For presentational purposes, we restrict ourselves to a bound on the variance. We refer to Appendix B for proofs.

Proposition 3 (Expected Value of $N_{\mathcal{L}'}$) *For $a, b \geq 1$, we have*

$$\begin{aligned}\mathbb{E}\left(N_{\mathcal{L}'}\right) &= \mathbb{E}\left(N_{\mathcal{L}'}^{(1)}\right) + \mathbb{E}\left(N_{\mathcal{L}'}^{(2)}\right) = a\mathbb{E}\left(\xi^{(b)}\right) + b\mathbb{E}\left(\eta^{(a)}\right) \\ &= a\frac{1-q^b}{p} + ab(q-q^b)q^{a-1} + b\sum_{k=1}^{a}\sum_{c=0}^{a-k} k\binom{c+k-1}{c}q^{cb}(q-q^b)^{k-1}p\,.\end{aligned}$$

Proposition 4 (Variance of $N_{\mathcal{L}'}$) *For $a, b \geq 1$, we have*

$$\mathbb{V}\mathrm{ar}\left(N_{\mathcal{L}'}\right) \leq \left[\sqrt{a\mathbb{V}\mathrm{ar}\left(\xi^{(b)}\right)} + \sqrt{H(a,b) - \left(\mathbb{E}\left(N_{\mathcal{L}'}\right) - a\frac{1-q^b}{p}\right)^2}\right]^2,$$

where $\mathbb{V}\mathrm{ar}\left(\xi^{(b)}\right)$ is provided in Equation (5)*, and $H(a,b)$ equals*

$$\begin{aligned}&a(a-1)b(q-q^b)^2q^{a-2} + ab(q-q^b)q^{a-1} + a(a-1)b(q^2-q^b)(q^2-q^b+2pq)q^{2(a-2)} \\ &\quad + ab(q^2-q^b+2pq)q^{2(a-1)} + 2ab(b-1)p^{-1}(1-q^{b-1})^2q^a \\ &\quad - 2ab(b-1)q^{2(a-2)}(q-q^b)\left[q^b + a(q^2-q^b) + (q^2-q^b)qp^{-1}\right] \\ &\quad + 2b(b-1)\sum_{k=2}^{a}\left[k(q-q^b)^k - k(q^2-q^b)^k - k^2pq(q^2-q^b)^{k-1}\right]\sum_{c=0}^{a-k}\binom{c+k-2}{c}q^{cb-1} \\ &\quad + b\sum_{k=1}^{a}k^2\left[(q^2-q^b)^{k-1}p^2 + (q-q^b)^{k-1}p\right]\sum_{c=0}^{a-k}\binom{c+k-1}{c}q^{cb}\,.\end{aligned}$$

4.3 Algorithm $\mathcal{L}''$

This section presents the complexity results for $\mathcal{L}''$. Again we use the representation $N_{\mathcal{L}''} = N_{\mathcal{L}''}^{(1)} + N_{\mathcal{L}''}^{(2)}$, where each component can be written as a sum of random variables as in Equation (11). The analysis is simpler than for $\mathcal{L}'$, since now all the $\gamma^{(b)}$ are mutually independent and identically distributed as $\xi^{(b)}$, and the $\eta^{(a)}$ are mutually independent and identically distributed as $\xi^{(a)}$. This observation allows to immediately write down the expected number of checks for $\mathcal{L}''$.

Proposition 5 (Expected Value of $N_{\mathcal{L}''}$)

$$\mathbb{E}\left(N_{\mathcal{L}''}\right) = \mathbb{E}\left(N_{\mathcal{L}''}^{(1)}\right) + \mathbb{E}\left(N_{\mathcal{L}''}^{(2)}\right) = a\mathbb{E}\left(\xi^{(b)}\right) + b\mathbb{E}\left(\xi^{(a)}\right) = a\frac{1-q^b}{p} + b\frac{1-q^a}{p}\,.$$

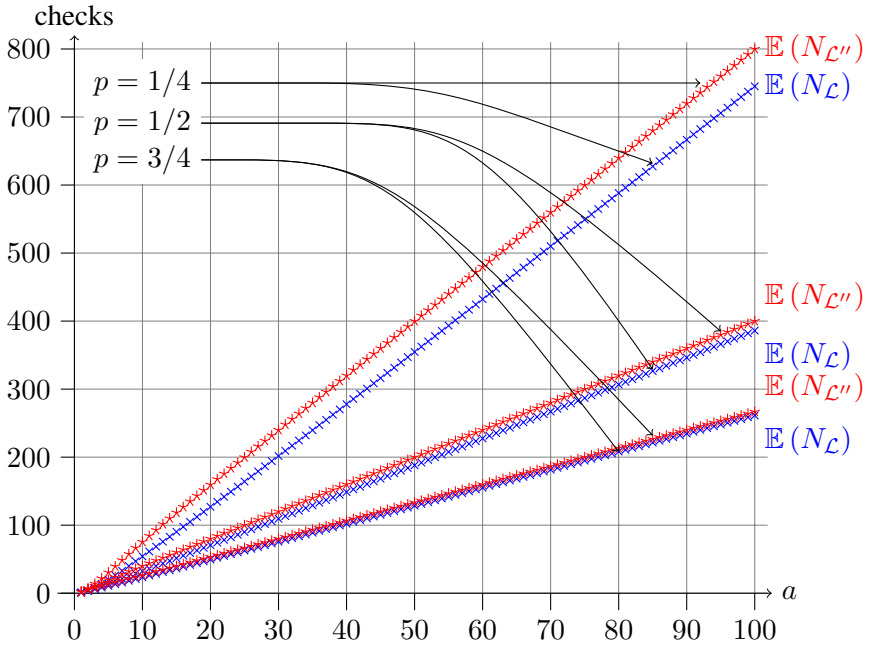

Figure 9: $\mathbb{E}\left(N_{\mathcal{L}}\right)$ and $\mathbb{E}\left(N_{\mathcal{L}''}\right)$ against $a = b$, for $p \in \{\, 1/4, 1/2, 3/4 \,\}$, and $1 \leq a = b \leq 100$.

Calculating the variance of $N_{\mathcal{L}''}$ is more difficult as $N_{\mathcal{L}''}^{(1)}$ and $N_{\mathcal{L}''}^{(2)}$ are dependent. The proof of the following proposition may be found in Appendix C.

Proposition 6 (Variance of $N_{\mathcal{L}''}$) *For $a, b \geq 1$, we have*

$$\mathbb{V}\mathrm{ar}\left(N_{\mathcal{L}''}\right) = a\mathbb{V}\mathrm{ar}\left(\xi^{(b)}\right) + b\mathbb{V}\mathrm{ar}\left(\xi^{(a)}\right) + 2J(a,b)\,,$$

where $\mathbb{V}\mathrm{ar}\left(\xi^{(\ell)}\right)$ is provided in Equation (5)*, and where*

$$J(a,b) = ab\frac{q^{a+b-1}}{p} - aq^a\frac{1-q^b}{p^2} - bq^b\frac{1-q^a}{p^2} + \frac{q(1-q^a)(1-q^b)}{p^3}\,.$$

5. Analysis of the Expectation and Variance of the Number of Checks

The aim of this section is twofold. The first is to quantitatively compare the expected running time of the three algorithms. This is done in Section 5.1. The second is to show that the distribution of the number of checks is, typically, sharply concentrated about its mean value. This is done in Section 5.2, relying on an analysis of the variances.

Before we start with our quantitative comparison it is useful to consider Figure 9, which depicts the expected values of the checks for $\mathcal{L}$ and $\mathcal{L}''$. The expected checks for $\mathcal{L}'$ are not shown since it is almost impossible to see the difference between $\mathbb{E}\left(N_{\mathcal{L}'}\right)$ and $\mathbb{E}\left(N_{\mathcal{L}''}\right)$. The expected values are plotted for $1 \leq a = b \leq 100$ and $p \in \{\, 1/4, 1/2, 3/4 \,\}$. The graphs of $\mathbb{E}\left(N_{\mathcal{L}}\right)$ are plotted with crosses. The graphs of $\mathbb{E}\left(N_{\mathcal{L}''}\right)$ are plotted with stars. It is almost impossible to spot the difference between the graphs of $\mathbb{E}\left(N_{\mathcal{L}}\right)$ and $\mathbb{E}\left(N_{\mathcal{L}''}\right)$ for $p = 3/4$. The trends of the graphs clearly indicate

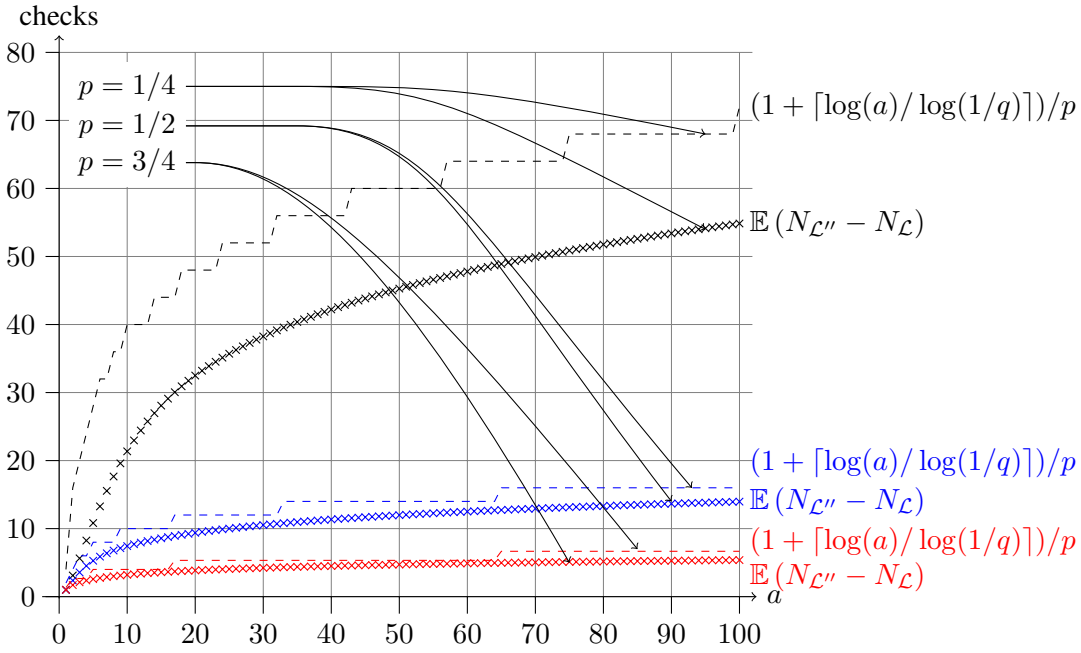

Figure 10: $\mathbb{E}\,(N_{\mathcal{L}''} - N_{\mathcal{L}})$ against $a = b$, for $p \in \{\,1/4, 1/2, 3/4\,\}$, and $1 \le a = b \le 100$. The function $\frac{1+\lceil \log(a)/\log(1/q)\,\rceil}{p}$ is plotted for reference purposes. The two black graphs at the top are for $p = 1/4$, the two blue graphs in the middle are for $p = 1/2$, and the two red graphs at the bottom are for $p = 3/4$.

that $\mathbb{E}\,(N_{\mathcal{L}}) < \mathbb{E}\,(N_{\mathcal{L}''})$. In the remainder of this section we shall provide a better understanding of this difference as well as how the underlying random quantities fluctuate for large a and b.

5.1 The Expected Number of Checks

In order to compare the expected number of checks for the three algorithms, we exploit that $\mathcal{L} \prec \mathcal{L}' \prec \mathcal{L}''$. It is thus of interest to derive lower and upper bounds for the difference $\mathbb{E}\,(N_{\mathcal{L}''}) - \mathbb{E}\,(N_{\mathcal{L}})$. A straightforward computation shows that

$$0 \le \mathbb{E}\,(N_{\mathcal{L}''}) - \mathbb{E}\,(N_{\mathcal{L}}) = \frac{1}{p} \sum_{\ell=0}^{b-1} \left(1 - (1 - pq^{\ell})^{a}\right) \le \frac{b}{p}\,. \tag{13}$$

The inequalities are easy since the summand is nonnegative and less than or equal to 1. The following presents a better upper bound on the difference of the expected values.

Proposition 7 (Upper Bound) *We have*

$$\mathbb{E}\,(N_{\mathcal{L}''}) - \mathbb{E}\,(N_{\mathcal{L}}) \le \frac{\min(1 + \lceil \ln(a)/\ln(1/q)\,\rceil\,, b)}{p}\,. \tag{14}$$

Proof Set $\ell_0 = \lceil \ln(a)/\ln(1/q)\,\rceil$. In view of Equation (13), it suffices to show that $\mathbb{E}\,(N_{\mathcal{L}''}) - \mathbb{E}\,(N_{\mathcal{L}}) \le (1 + \ell_0)/p$. For this, we use the fact that $\ell \ge \ell_0$ implies $aq^{\ell} \le 1$. This suggests using the

Bernoulli Inequality, which implies that

$$(1+x)^r \geq 1 + rx\,, \qquad \text{for } r \geq 1 \text{ and } x \geq -1\,.$$

Using this inequality we note that

$$\sum_{\ell=0}^{b-1}(1-pq^\ell)^a \geq \sum_{\ell=\ell_0}^{b-1}(1-pq^\ell)^a \geq \sum_{\ell=\ell_0}^{b-1}\left(1-apq^\ell\right) = b-\ell_0-a(q^{\ell_0}-q^b) \geq b-\ell_0-1\,,$$

where the last inequality follows the fact that $aq^{\ell_0} \leq 1$. This lower bound can be used in conjunction with the equality in Equation (13) to prove the desired upper bound. ■

Figure 10 plots the graph of $\mathbb{E}\left(N_{\mathcal{L}''} - N_{\mathcal{L}}\right)$ against $a = b$ for $p \in \{\,1/4, 1/2, 3/4\,\}$. For reference purposes the graphs of the function $\left(1 + \lceil \log(a)/\log(1/q) \rceil\right)/p$ are also shown to allow comparison with Equation (14). The graphs of $\mathbb{E}\left(N_{\mathcal{L}''} - N_{\mathcal{L}}\right)$ are plotted with crosses. The graphs of the reference functions are plotted with dashes. It may be deduced from Figure 10 that the bound from Equation (14) is pretty accurate.

5.2 The Variance of the Number of Checks

This section shows that, by analysing the variances, the number of checks is likely to be close to its expected value for large a and b.

This is formalised by setting $b = \lfloor \beta a \rfloor$ for some constant $\beta > 0$, and by studying the expected values and variances as $a \to \infty$. Proposition 7 shows that the expected values are of order $a + b$, which is of order a. Knowledge about the order of the variances leads to powerful statements, since for any $\mathcal{A} \in \{\,\mathcal{L}, \mathcal{L}', \mathcal{L}''\,\}$ and for large x it is highly likely that $N_{\mathcal{A}}$ lies in the interval

$$\left[\,\mathbb{E}\left(N_{\mathcal{A}}\right) - x\sqrt{\mathbb{V}\mathrm{ar}\left(N_{\mathcal{A}}\right)}, \mathbb{E}\left(N_{\mathcal{A}}\right) + x\sqrt{\mathbb{V}\mathrm{ar}\left(N_{\mathcal{A}}\right)}\,\right],$$

To see this, note that by the Chebyshev Inequality, for $x > 0$,

$$\mathbb{P}\left(\left|N_{\mathcal{A}} - \mathbb{E}\left(N_{\mathcal{A}}\right)\right| > x\sqrt{\mathbb{V}\mathrm{ar}\left(N_{\mathcal{A}}\right)}\right) \leq \frac{1}{x^2}\,.$$

Our expressions for the variances thus give intervals in which $N_{\mathcal{A}}$ lies with probability of at least $1 - x^{-2}$.

Let us start with the order of the variance for Algorithm $\mathcal{L}''$. For this we use Proposition 6. With the above definitions of a and b, it is easy to see that $J(a,b)$ tends to a constant as $a \to \infty$, and with Equation (4) we see that $\mathbb{V}\mathrm{ar}\left(\xi^{(a)}\right)$ and $\mathbb{V}\mathrm{ar}\left(\xi^{(b)}\right)$ tend to qp^{-2}. Conclude that, as $a \to \infty$, $\mathbb{V}\mathrm{ar}\left(N_{\mathcal{L}''}\right)$ behaves like $q(a+b)/p^2$, which is of the order a. Therefore, the distribution of $N_{\mathcal{L}''}$ is mostly concentrated on an interval of length $\mathcal{O}\left(\sqrt{a}\right)$.

We next discuss the Algorithms $\mathcal{L}$ and $\mathcal{L}'$. Since these algorithms never carry out more checks than $\mathcal{L}''$, it follows that for $\mathcal{A} \in \{\mathcal{L}, \mathcal{L}'\}$ we have:

$$\begin{aligned}
\mathbb{V}\mathrm{ar}\left(N_{\mathcal{A}}\right) &\leq \mathbb{E}\left(N_{\mathcal{L}''}^2\right) - \left(\mathbb{E}\left(N_{\mathcal{A}}\right)\right)^2 \\
&\leq \mathbb{E}\left(N_{\mathcal{L}''}^2\right) - \left(\mathbb{E}\left(N_{\mathcal{L}''}\right) - \frac{1 + \lfloor \ln(a)/\ln(1/q) \rfloor}{p}\right)^2 \\
&\leq \mathbb{V}\mathrm{ar}\left(N_{\mathcal{L}''}\right) + 2\mathbb{E}\left(N_{\mathcal{L}''}\right)\frac{1 + \lfloor \ln(a)/\ln(1/q) \rfloor}{p}\,,
\end{aligned}$$

where the second inequality follows from Proposition 7. The first term is of order a, as shown already. The second term is of order $a \ln a$. For $\mathcal{L}$ and $\mathcal{L}'$ the expected value is thus of order a while the distribution is mostly concentrated on an interval of length $\mathcal{O}\left(\sqrt{a \ln a}\right)$ about the mean.

6. An Upper Bound on the Expected Time Complexity of AC-3

In this section we shall use two key observations, which allow us to derive an upper bound for the expected time complexity of AC-3. We shall assume that the arcs are selected from the queue in first-in-first-out order.

Let $N_{\mathcal{R}}$ denote the number of checks which are required by *revise* (Mackworth, 1977) for a random $a \times b$ constraint. It follows from Proposition 3 that $\mathbb{E}(N_{\mathcal{R}}) = a(1 - q^b)/p$. During the execution of AC-3 the algorithm *revise* is applied several times. Our assumption that a random check succeeds with a uniform probability of p is correct if the pair of variables which are involved in a revision are involved in a revision together for the first time. In all other cases we may assume that if the maximum domain size is d then $d(1 - q^d)/p$ is an *upper bound* for the expected number of checks which are required by *revise* for *any* revision. This is the first key observation.

The second key observation is that if $D(v)$ is revised against $D(w)$ for the i-th time then $1 \leq |\, D(w)\,| \leq d + 1 - i$, where d is its initial domain size.

These are the keys to the following proposition, which is our final result.

Proposition 8 (Expected Time Complexity of AC-3 with Maximal Revisions) *Let $N_{\mathcal{A}_3}$ denote the checks which are required by* AC-3 *when applied to a random* CSP *which has no unconstrained variables. Assuming that the arcs are taken in a first-in-first-out order from the queue, then*

$$\mathbb{E}(N_{\mathcal{A}_3}) \leq \frac{4ed + (2e - n)d(d-1)}{2p}\,.$$

Proof It follows from the analysis by Mackworth and Freuder (1985) that the worst possible time complexity for AC-3 occurs if the domains of each of the variables are revised d times against the domains of each of the variables by which they are constrained. This gives rise to an initial cycle consisting of $2e$ revisions and $d - 1$ remaining cycles each of which consisting of $2e - n$ revisions. Let $1 \leq i \leq d$ and let $\ell = d + 1 - i$. In the i-th cycle the size of a domain is in $\{\, \ell - 1, \ell \,\}$. Therefore, the cost of any revision in the i-th cycle cannot exceed $\ell(1 - q^\ell)/p$. As a consequence we have

$$\mathbb{E}(N_{\mathcal{A}_3}) \leq 2ed\frac{1 - q^d}{p} + \sum_{\ell=1}^{d-1}(2e - n)\ell\frac{1 - q^\ell}{p} \leq \frac{4ed + (2e - n)d(d-1)}{2p}\,,$$

which completes the proof. ■

It is interesting to notice that (for constant p) the upper bound on the expected time complexity of the non-optimal algorithm AC-3 is $\mathcal{O}\left(ed^2\right)$, which is the best possible worst-case time complexity.

7. Conclusions

This paper analyses three revision algorithms for 2-variable Constraint Satisfaction Problems (CSPs). The algorithms are called $\mathcal{L}$, $\mathcal{L}'$, and $\mathcal{L}''$. For 2-variable CSPs $\mathcal{L}$ models an arc consistency algorithm

which exploits multi-directionality. However, $\mathcal{L}'$ and $\mathcal{L}''$ do not exploit multi-directionality. For 2-variable CSPs $\mathcal{L}'$ models AC-3. The results are parametrised over the probability p that a random check succeeds and probability $q = 1 - p$ that it fails. Our results demonstrate that the average time complexity of $\mathcal{L}''$ is given by $[a - aq^b + b - bq^a]/p$, where a and b are the domain sizes of the variables. The expected time complexity of $\mathcal{L}$ is $\sum_{\ell=0}^{b-1}[\,1 - (1 - pq^\ell)^a\,]/p$ below that of $\mathcal{L}''$. It is proved that $0 \leq \sum_{\ell=0}^{b-1}[\,1 - (1 - pq^\ell)^a\,]/p \leq \min(b, 1 + \lceil\, \ln(a)/\ln(1/q)\,\rceil)/p$. It is shown that $\mathbb{E}\,(N_{\mathcal{L}}) < \mathbb{E}\,(N_{\mathcal{L}'}) < \mathbb{E}\,(N_{\mathcal{L}''})$. It is proved that, as the domain sizes a and b become large, the number of checks which are required by the algorithms is sharply concentrated about their expected number of checks. An upper bound of $2ed/p + (2e - n)d(d-1)/2p$ is presented for the expected time complexity of AC-3 for the general binary CSP.

Acknowledgments

The authors thank Christophe Lecoutre and Deepak Mehta for comments and suggestions. This work has received support from the Irish Research Council for Science, Engineering and Technology. The last two authors have received support from the Irish Science Foundation under Grant SFI04/RP1/I515.

Appendix A. Proofs for Algorithm $\mathcal{L}$

In this appendix we shall prove Propositions 1 and 2.

A.1 The First Moment of $N_{\mathcal{L}}$

In this part of the appendix we shall derive the expected value of of $N_{\mathcal{L}}$, which is stated in Proposition 1. To prove the proposition we evaluate the derivative of the generating function in $z = 1$. We stress that $\mathbb{E}\,(N_{\mathcal{L}})$ depends on a and b.

In terms of the derivative ∂f_1 of f_z in $z = 1$, Equation (7) yields

$$\mathbb{E}\,(N_{\mathcal{L}}) = \sum_{c=0}^{a} \binom{a}{c} q^{b(a-c)} \left(\frac{p}{q}\right)^c [\,b(a-c)f_1(b,c) + \partial f_1(b,c)\,]\ .$$

Applying the multinomial theorem to Equation (8) for $z = 1$, one sees that

$$f_1(b,c) = \left(\sum_{\ell=1}^{b} q^\ell\right)^c = \left(\frac{q - q^{b+1}}{p}\right)^c. \tag{15}$$

Using the identity

$$\sum_{c=0}^{a} c\binom{a}{c} x^c y^{a-c} = ax(x+y)^{a-1}, \tag{16}$$

we see that the proposition is proved once we show that

$$\sum_{c=0}^{a} \binom{a}{c} q^{b(a-c)} \left(\frac{p}{q}\right)^c \partial f_1(b,c) = F(a,b). \tag{17}$$

For this, we use the following lemma; it is also a key ingredient in the next section.

Lemma 9 *We have $\partial f_1(1,c) = cq^c$ for $c \geq 0$, and for $b > 1, c \geq 0$ we have*

$$\partial f_1(b,c) = c\left(\frac{q}{p}\right)^c \left(1-q^b\right)^{c-1} \frac{1-q^b-bpq^b}{p} + \frac{1}{p}\left(\frac{q}{p}\right)^c \sum_{\ell=0}^{b-1} \left(1-q^b-pq^\ell\right)^c - \frac{b}{p}\left(\frac{q}{p}\right)^c \left(q-q^b\right)^c . \tag{18}$$

Proof The formula for $\partial f_1(1,c)$ follows from the observation that $f_z(1,c) = (qz)^c$, so we may focus on $b > 1$. The recursion Equation (9) for f_z yields for $b > 1$, $c \geq 0$,

$$\partial f_1(b,c) = g(b,c) + \sum_{\ell=0}^{c} \binom{c}{\ell} q^{b\ell} \partial f_1(b-1, c-\ell), \tag{19}$$

where, using Equation (3),

$$\begin{aligned} g(b,c) &= \frac{1-q^c}{p}\left(\frac{q-q^b}{p}\right)^c + b\sum_{\ell=1}^{c} \ell\binom{c}{\ell} q^{b\ell}\left(\frac{q-q^b}{p}\right)^{c-\ell} \\ &= \frac{1-q^c}{p}\left(\frac{q-q^b}{p}\right)^c + bcq^b\left(\frac{q-q^{b+1}}{p}\right)^{c-1} . \end{aligned}$$

A mathematical induction argument shows that it suffices to calculate the right-hand side with Equation (18) for $b-1$, and then observe that this yields the above expression for Equation (18) for b. This is readily verified using Newton's Binomial Theorem and Equation (16). ■

To prove Equation (17), we apply the preceding lemma to rewrite the left-hand side. We find with Equation (16) that

$$\sum_{c=0}^{a} \binom{a}{c} cq^{b(a-c)}(1-q^b)^{c-1} \frac{1-q^b-bpq^b}{p} = a\frac{1-q^b-bpq^b}{p} .$$

On the other hand, Newton's Binomial Theorem yields

$$\sum_{c=0}^{a} \binom{a}{c} q^{b(a-c)} \frac{1}{p} \sum_{\ell=0}^{b-1} \left(1-q^b-pq^\ell\right)^c = \frac{1}{p}\sum_{\ell=0}^{b-1}(1-pq^\ell)^a$$

and

$$\sum_{c=0}^{a} \binom{a}{c} q^{b(a-c)} \frac{b}{p}(q-q^b)^c = \frac{bq^a}{p} .$$

This proves Equation (17) and thus the proposition.

A.2 The Second Moment of $\mathcal{L}$

In this part of the appendix we shall derive the expected value of $N_{\mathcal{L}}^2$, which is needed for the derivation of $\mathbb{V}\mathrm{ar}\,(N_{\mathcal{L}})$. Using Equation (2) and proceeding in a similar way as in the previous

section, this yields

$$\begin{aligned}\mathbb{E}\left(N_{\mathcal{L}}^2\right) &= \mathbb{E}\left(N_{\mathcal{L}}\right) + \sum_{c=0}^{a}\binom{a}{c} b(a-c)[b(a-c)-1]q^{b(a-c)}(1-q^b)^c \\ &\quad + 2\sum_{c=0}^{a}\binom{a}{c} b(a-c)q^{b(a-c)}\left(\frac{p}{q}\right)^c \partial f_1(b,c) \\ &\quad + \sum_{c=0}^{a}\binom{a}{c} q^{b(a-c)}\left(\frac{p}{q}\right)^c \partial^2 f_1(b,c)\,. \end{aligned} \tag{20}$$

The expected value of $N_{\mathcal{L}}$ has already been determined in the previous section, and the first sum equals $a(a-1)b^2q^{2b} + ab(b-1)q^b$ as readily verified with the identity

$$\sum_{c=0}^{a} c^2\binom{a}{c}x^c y^{a-c} = a(a-1)x^2(x+y)^{a-2} + ax(x+y)^{a-1}. \tag{21}$$

To calculate the second sum, we note that

$$\begin{aligned}&\sum_{c=0}^{a}\binom{a}{c}(a-c)q^{b(a-c)}\left(\frac{p}{q}\right)^c \partial f_1(b,c) \\ &= \sum_{c=1}^{a}\binom{a}{c}cq^{bc}\left(\frac{p}{q}\right)^{a-c} \partial f_1(b,a-c) \\ &= aq^b\sum_{c=0}^{a-1}\binom{a-1}{c}q^{bc}\left(\frac{p}{q}\right)^{a-1-c} \partial f_1(b,a-1-c) \\ &= aq^b\sum_{c=0}^{a-1}\binom{a-1}{c}q^{b(a-1-c)}\left(\frac{p}{q}\right)^c \partial f_1(b,c) \\ &= aq^b F(a-1,b)\,, \end{aligned} \tag{22}$$

and $F(a-1,b)$ has already been found in Equation (10). To make this expression also valid for $a=1$, we need to interpret $F(0,b)$ as zero.

Let us write

$$G(a,b) = \sum_{c=0}^{a}\binom{a}{c}q^{b(a-c)}\left(\frac{p}{q}\right)^c \partial^2 f_1(b,c)\,,$$

so that

$$\mathbb{E}\left(N_{\mathcal{L}}^2\right) = a(a-1)b^2q^{2b} + ab^2q^b + F(a,b) + 2abq^bF(a-1,b) + G(a,b)\,.$$

Since $\partial^2 f_1(1,c) = c(c-1)q^c$, we see with Equation (21) that $G(a,1) = a(a-1)p^2$. We next derive a recursion for the last sum in Equation (20), which allows us to find $G(a,b)$ explicitly.

For this, we use the following consequence of Equation (9): for $b>1$, $\partial^2 f_1(b,c)$ is given by

$$\begin{aligned}&\sum_{\ell=0}^{c}\binom{c}{\ell}q^{b\ell}\left[\partial^2 f_1(b-1,c-\ell) + 2b\ell\partial f_1(b-1,c-\ell) + b\ell\left(b\ell-1\right)f_1(b-1,c-\ell)\right] \\ &\quad + 2\mathbb{E}\left(\xi^{(c)}\right)\partial f_1(b-1,c) + \mathbb{E}\left(\xi^{(c)}\left[\xi^{(c)}-1\right]\right)f_1(b-1,c)\,. \end{aligned}$$

Note that, by definition, to find G we need to sum this appropriately over c. For easy reference, let us refer to each of the five terms by a roman letter, i.e., $\partial^2 f_1(b,c) = \mathrm{I} + \mathrm{II} + \mathrm{III} + \mathrm{IV} + \mathrm{V}$. We start by calculating the sum over c for I:

$$\sum_{c=0}^{a} \binom{a}{c} q^{b(a-c)} \left(\frac{p}{q}\right)^c \sum_{\ell=0}^{c} \binom{c}{\ell} q^{b(c-\ell)} \partial^2 f_1(b-1,\ell)\,.$$

Using $\sum_{c=0}^{a}\sum_{\ell=0}^{c} = \sum_{\ell=0}^{a}\sum_{c=\ell}^{a}$, we next rewrite the double sum by shifting the index c over ℓ, yielding

$$\sum_{\ell=0}^{a}\sum_{c=0}^{a-\ell} \binom{a}{c+\ell}\binom{c+\ell}{\ell} q^{b(a-\ell)} \left(\frac{p}{q}\right)^{c+\ell} \partial^2 f_1(b-1,\ell)\,.$$

After noting that $\binom{a}{c+\ell}\binom{c+\ell}{\ell} = \binom{a-\ell}{c}\binom{a}{\ell}$, we can calculate the sum over c with the binomial theorem. This shows that the sum over I equals

$$\sum_{\ell=0}^{a} \binom{a}{\ell} q^{(b-1)(a-\ell)} \left(\frac{p}{q}\right)^{\ell} \partial^2 f_1(b-1,\ell) = G(a,b-1)\,.$$

A similar argument allows for finding the sum over II:

$$\begin{aligned}
&2\sum_{c=0}^{a} \binom{a}{c} q^{b(a-c)} \left(\frac{p}{q}\right)^c \sum_{\ell=0}^{c} \binom{c}{\ell} q^{b\ell} b\ell \partial f_1(b-1,c-\ell) \\
&\quad = 2p\sum_{\ell=0}^{a} \binom{a}{\ell} q^{(b-1)(a-\ell)} \left(\frac{p}{q}\right)^{\ell} b(a-\ell)\partial f_1(b-1,\ell) \\
&\quad = 2abpq^{b-1}F(a-1,b-1)\,,
\end{aligned}$$

where the last equality follows from Equation (22).

We next consider the sum over III. With formula Equation (15) for f_1, we obtain

$$\begin{aligned}
&\sum_{\ell=0}^{c} \binom{c}{\ell} q^{b\ell} b\ell(b\ell-1) f_1(b-1,c-\ell) \\
&\quad = b^2c(c-1)\left(\frac{p}{q}\right)^{2-c} q^{2b}(1-q^b)^{c-2} + b(b-1)c\left(\frac{p}{q}\right)^{1-c} q^b(1-q^b)^{c-1}\,,
\end{aligned}$$

so that

$$\begin{aligned}
&\sum_{c=0}^{a} \binom{a}{c} q^{b(a-c)} \left(\frac{p}{q}\right)^c \sum_{\ell=0}^{c} \binom{c}{\ell} q^{b\ell} b\ell(b\ell-1) f_1(b-1,c-\ell) \\
&\quad = a(a-1)b^2p^2q^{2b-2} + ab(b-1)pq^{b-1}\,.
\end{aligned}$$

As for the sum over IV, we conclude with Lemma 9 and some calculations that

$$\begin{aligned}
&2\sum_{c=0}^{a}\binom{a}{c}q^{b(a-c)}\left(\frac{p}{q}\right)^{c}\mathbb{E}\left(\xi^{(c)}\right)\partial f_1(b-1,c)\\
&=2a\frac{1-q^{b-1}-(b-1)pq^{b-1}}{p^2}(1-pq^{b-1})^{a-1}+\frac{2}{p^2}\sum_{\ell=0}^{b-2}(1-pq^{b-1}-pq^{\ell})^{a}\\
&\quad-2\frac{b-1}{p^2}(q-pq^{b-1})^{a}-\frac{2q^{a}F(a,b-1)}{p}\,.
\end{aligned}$$

Finally, for V we use Equation (15), Equation (3) and Equation (4) to obtain that

$$\begin{aligned}
&\sum_{c=0}^{a}\binom{a}{c}q^{b(a-c)}\left(\frac{p}{q}\right)^{c}\mathbb{E}\left(\xi^{(c)}\left[\,\xi^{(c)}-1\,\right]\right)f_1(b-1,c)\\
&\quad=\frac{2q}{p^2}(1-pq^{b-1})^{a}-\frac{2q^{a+1}}{p^2}-\frac{2}{p}a(1-q^{b-1})q^{a}\,.
\end{aligned}$$

Clearly, we have for $a,b\geq 1$,

$$G(a,b)=G(a,1)+\sum_{\ell=2}^{b}\left[\,G(a,\ell)-G(a,\ell-1)\,\right]\,.$$

The summand is II + III + IV + V with $b=\ell$, which is exactly $h(\ell)$.

Appendix B. Proofs for Algorithm $\mathcal{L}'$

In this appendix we shall prove Propositions 3 and 4. For Proposition 3, it suffices to compute $\mathbb{E}\left(\eta^{(a)}\right)$ with (12) and (16).

The inequality stated in Proposition 4 is derived from the Cauchy-Schwarz inequality

$$\mathbb{V}\mathrm{ar}\left(N_{\mathcal{L}'}\right)\leq\left(\sqrt{\mathbb{V}\mathrm{ar}\left(N_{\mathcal{L}'}^{(1)}\right)}+\sqrt{\mathbb{V}\mathrm{ar}\left(N_{\mathcal{L}'}^{(2)}\right)}\right)^{2}.$$

Since the $\gamma^{(b)}$ are independent and identically distributed according to the distribution of $\xi^{(b)}$, we have $\mathbb{V}\mathrm{ar}\left(N_{\mathcal{L}'}^{(1)}\right)=a\mathbb{V}\mathrm{ar}\left(\xi^{(b)}\right)$. To finish the proof, we need to find $H(a,b)=\mathbb{E}\left(\left(N_{\mathcal{L}'}^{(2)}\right)^{2}\right)$.

The main tool is the identity (for any $i\neq j$)

$$\mathbb{E}\left(\left(N_{\mathcal{L}'}^{(2)}\right)^{2}\right)=b(b-1)\mathbb{E}\left(\eta_i^{(a)}\eta_j^{(a)}\right)+b\mathbb{E}\left(\left(\eta_i^{(a)}\right)^{2}\right).$$

In view of (12), the last term is readily found to be

$$a(a-1)b(q-q^{b})^{2}q^{a-2}+ab(q-q^{b})q^{a-1}+b\sum_{k=1}^{a}\sum_{c=0}^{a-k}k^{2}\binom{c+k-1}{c}q^{cb}(q-q^{b})^{k-1}p,$$

and we next focus on the calculation of $\mathbb{E}\left(\eta_i^{(a)}\eta_j^{(a)}\right)$ for $i \neq j$. First observe that

$$\mathbb{E}\left(\eta_i^{(a)}\eta_j^{(a)}\right) = 2\sum_{\substack{\ell,k=1\\k<\ell}}^{a} k\ell\mathbb{P}\left(\eta_i^{(a)}=k,\eta_j^{(a)}=\ell\right) + \sum_{\ell=1}^{a}\ell^2\mathbb{P}\left(\eta_i^{(a)}=\ell,\eta_j^{(a)}=\ell\right). \tag{23}$$

We use similar arguments as those leading to (12) to see that for $i \neq j$, $k < \ell$,

$$\mathbb{P}\left(\eta_i^{(a)}=k,\eta_j^{(a)}=\ell\right) = \binom{a}{\ell}q^{b(a-\ell)}(q-q^b)^{\ell-k}pq(q^2-q^b)^{k-1} + (q^2-q^b)^{k-1}(q-q^b)^{\ell-k-1}p^2q\sum_{c=0}^{a-\ell}\binom{c+\ell-2}{c}q^{cb},$$

while

$$\mathbb{P}\left(\eta_i^{(a)}=\ell,\eta_j^{(a)}=\ell\right) = \binom{a}{\ell}q^{b(a-\ell)}(q^2-q^b)^{\ell-1}(q^2-q^b+2pq)+(q^2-q^b)^{\ell-1}p^2\sum_{c=0}^{a-\ell}\binom{c+\ell-1}{c}q^{cb}.$$

Let us next study the first term in (23). After first calculating the sum over k, we find after some tedious computations that

$$\begin{aligned}
&\sum_{\substack{\ell,k=1\\k<\ell}}^{a} k\ell\mathbb{P}\left(\eta_i^{(a)}=k,\eta_j^{(a)}=\ell\right)\\
&\quad= \sum_{\ell=2}^{a}\left[\,\ell(q-q^b)^\ell - \ell^2pq(q^2-q^b)^{\ell-1} - \ell(q^2-q^b)^\ell\,\right]\sum_{c=0}^{a-\ell}\binom{c+\ell-2}{c}q^{cb-1}\\
&\qquad + ap^{-1}(1-q^{b-1})^2q^a - aq^{2a-4}(q-q^b)\left[\,a(q^2-q^b)+q^b+(q^2-q^b)qp^{-1}\,\right].
\end{aligned}$$

Similarly, we find

$$\begin{aligned}
&\sum_{\ell=1}^{a}\ell^2\mathbb{P}\left(\eta_i^{(a)}=\ell,\eta_j^{(a)}=\ell\right)\\
&\quad= a(a-1)(q^2-q^b)(q^2-q^b+2pq)q^{2(a-2)} + a(q^2-q^b+2pq)q^{2(a-1)}\\
&\qquad + \sum_{\ell=1}^{a}\ell^2(q^2-q^b)^{\ell-1}p^2\sum_{c=0}^{a-\ell}\binom{c+\ell-1}{c}q^{cb}.
\end{aligned}$$

Appendix C. Proofs for Algorithm $\mathcal{L}''$

In this appendix we shall prove Proposition 6 only, as Proposition 5 has already been proven.

In order to prove Proposition 6, it suffices to show that

$$\mathbb{E}\left(N_{\mathcal{L}''}^{(1)}N_{\mathcal{L}''}^{(2)}\right) = J(a,b) + ab\frac{(1-q^a)(1-q^b)}{p^2}, \tag{24}$$

To this end, we first write

$$\mathbb{E}\left(N^{(1)}_{\mathcal{L}''}N^{(2)}_{\mathcal{L}''}\right)=\sum_{i=1}^{a}\sum_{j=1}^{b}\mathbb{E}\left(\gamma_i^{(b)}\eta_j^{(a)}\right)\,.$$

We calculate the summand by writing, for $i<a$ and $j<b$,

$$\mathbb{E}\left(\gamma_i^{(b)}\eta_j^{(a)}\right)=\mathbb{E}\left(\gamma_i^{(b)}\eta_j^{(a)}\left\{1_{\{\gamma_i^{(b)}<j\}}+1_{\{\gamma_i^{(b)}=j\}}+1_{\{\gamma_i^{(b)}>j\}}\right\}\right)\,.$$

Next note that

$$\mathbb{E}\left(\gamma_i^{(b)}\eta_j^{(a)}1_{\{\gamma_i^{(b)}<j\}}\right)=\mathbb{E}\left(\gamma_i^{(b)}1_{\{\gamma_i^{(b)}<j\}}\right)\mathbb{E}\left(\eta_j^{(a)}\right)=\frac{1-q^a}{p}\left(\frac{1-q^{j-1}}{p}-(j-1)q^{j-1}\right)\,,$$

while

$$\mathbb{E}\left(\gamma_i^{(b)}\eta_j^{(a)}1_{\{\gamma_i^{(b)}=j\}}\right)=jpq^{j-1}\left[\mathbb{E}\left(\eta_j^{(a)}1_{\{\eta_j^{(a)}<i\}}\right)+i\mathbb{P}\left(\eta_j^{(a)}=i\right)\right]=jq^{j-1}(1-q^i)$$

and

$$\begin{aligned}\mathbb{E}\left(\gamma_i^{(b)}\eta_j^{(a)}1_{\{\gamma_i^{(b)}>j\}}\right)&=\mathbb{E}\left(\gamma_i^{(b)}1_{\{\gamma_i^{(b)}>j\}}\right)\left[\mathbb{E}\left(\gamma^{(a-1)}1_{\{\gamma^{(a-1)}<i\}}\right)+\mathbb{E}\left((1+\gamma^{(a-1)})1_{\{\gamma^{(a-1)}\geq i\}}\right)\right]\\&=\left(\frac{q^j-q^b}{p}+jq^j\right)\left(\frac{1-q^{a-1}}{p}+q^{i-1}\right)\,.\end{aligned}$$

We have thus derived an expression for $\mathbb{E}\left(\gamma_i^{(b)}\eta_j^{(a)}\right)$ which is valid for $i<a$ and $j<b$. It can be seen that this expression also gives the correct answer in the complementary case, i.e., if either $i=a$ or $j=b$. We obtain Equation (24) by calculating the sum over $1\leq i\leq a$ and $1\leq j\leq b$.

References

C. Bessière and M. Cordier. Arc-consistency and arc-consistency again. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, 1993.

C. Bessière and J.-C. Régin. Refining the basic constraint propagation algorithm. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 309–315, 2001.

C. Bessière and J.-C. Régin. An arc-consistency algorithm optimal in the number of constraint checks. In M. Meyer, editor, *Proceedings of the ECAI'94 Workshop on Constraint Processing*, Amsterdam, 1994.

C. Bessière, E.C. Freuder, and J.-C. Régin. Using inference to reduce arc consistency computation. In C.S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 1, pages 592–598, Montréal, Québec, Canada, 1995. Morgan Kaufmann Publishers.

I.P. Gent, E. MacIntyre, P. Prosser, B. Smith, and T. Walsh. Random constraint satisfaction: Flaws and structure. *Journal of Constraints*, 6(4):345–372, 2001.

R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1989. ISBN 0-201-14236-8.

C. Lecoutre and F. Hemery. A study of residual supports in arc consistency. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 125–130, 2007.

C. Lecoutre, F. Boussemart, and F. Hemery. Exploiting multidirectionality in coarse-grained arc consistency algorithms. In F. Rossi, editor, *Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming, CP'2003*, pages 480–494, 2003.

A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

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.

R. Mohr and T. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.

R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Publishing Company, 1996. ISBN 0-201-40009-X.

M.R.C. van Dongen. Domain-heuristics for arc-consistency algorithms. In K.R. Apt, F. Fages, E.G. Freuder, B. O'Sullivan, F. Rossi, and T. Walsh, editors, *ERCIM/Colognet Workshop*, pages 72–83, 2002. URL `http://csweb.ucc.ie/~dongen/papers/ERCIM/02/ercim02.pdf`.

M.R.C. van Dongen. Saving support-checks does not always save time. *Artificial Intelligence Review*, 21(3–4), 2004.

R.J. Wallace. Why AC-3 is almost always better than AC-4 for establishing arc consistency in CSPs. In R. Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 239–245, 1993.

H.S. Wilf. *Generatingfunctionology*. Academic Press, Incorporated, 1994. ISBN 0-12-751956-4.

Y. Zhang and R.H.C. Yap. Making AC-3 an optimal algorithm. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 316–321, 2001.