Erratum
Constraint Networks

Christophe lecoutre

ISTE/Wiley

August 21, 2024

1. An obvious bad use of "copy and paste". Definition 3.28, Page 151, replace every occurrence of generalized arc-inconsistent by generalized arc-consistent. You should obtain:

   DEFINITION 3.28.-*[Generalized Arc Consistency]*

   - A constraint c is *generalized arc-consistent*, or *GAC-consistent* iff $\forall x \in scp(c), \forall a \in dom(x)$, there exists a support for (x, a) on c.
   - A constraint network P is *generalized arc-consistent* iff every constraint of P is generalized arc-consistent.

   Additionally, we have for any constraint network P:

   - A v-value (x, a) of P is *generalized arc-consistent* on P iff for every constraint c of P involving x, there is a support for (x, a) on c.
   - A variable x of P is *generalized arc-consistent* on P iff $\forall a \in dom(x)$, (x, a) is generalized arc-consistent on P.

2. Caption of Figure 9.2, Page 396. Replace FC-*dom* by FC-*max-dom*.

3. Line 6, Paragraph2, Page 284. Replace $O(nd)$ by $O(nr)$.

4. In order to avoid the use of an undefined value for $sup[c, x, a]$ at Line 17 of Algorithm 20 Page 211, we have to insert:

   foreach constraint c of P **do**

   between line 4 and line 5 of the same algorithm. The new loop must start at the first level (most extern level), meaning that the loop starting at line 2 is finished when starting the new loop. This has no effect on complexities. Thank you to Yves Deville for pointing this to me.

5. I do not know why I forgot to cite [BES 96] when introducing the heuristic *dom/ddeg* Page 394 (second paragraph).

6. Page 197, in Proposition 4.1, if we assume that the (binary) constraint network is normalized, the worst-case time complexity can be reduced to O(nd) since the last instantiated variable is involved in at most $n - 1$ binary constraints. Thank you to Liang Zhang from JiLin University for pointing this to me.

Additional remarks elaborated from a message nicely sent by Donald Knuth in December 2022:

7. Page 34, Fig 5. There should be no edge between x_2 and x_4.

8. Last sentence of Page 54 (after Definition 1.28). Although one could envision a more general definition, here, the context is that of a binary constraint network. Hence, the sentence should be: "**In a binary constraint network**, two v-values (x, a) and (y, b) are sais to be *compatible* if ...".

9. Pages 86-90, Section 1.5.1. An important missing reference is that of **dancing links**. Indeed, the underlying technique used behind the representation of domains corresponds to "dancink links", as described in:

   - D. Knuth. Dancing Links. Section 7.2.2.1 of Volume 4B of the Art of Computer Programming.
   - D. Knuth. Dancing Links, Millenial Perspectives in Computer Science, 2000, 187–214, `https://arxiv.org/pdf/cs/0011047`

   Dancing links are also used in the paper "C. Lecoutre and R. Szymanek: Generalized Arc Consistency for Positive Table Constraints. CP 2006: 284-298", but by the time we wrote this paper, we were not aware of the reference to "dancing links". In both the CP'06 paper and the book (page 200), we use the fact that "if $\tau[y] \notin dom(y)$ then $dom(y).next[\tau[y]]$ is less than or equal to the smalest value of $dom(y)$ strictly greater than $\tau[y]$. Donald Knuth told me that the proof is absent, but can be certainly derived from facts from exercises 7.2.2.1–1 and 7.2.2.1–2 of Section 7.2.2.1 of Volume 4B of the Art of Computer Programming.

10. Page 123, Section 2.2.5.1.2, Haystacks. Some details are given in Exercice 340 (the number of the exercice may change) of Section 7.2.2.3 of Volume 4B of the Art of Computer Programming.

11. Page 193, Algorithm 9. In case some unary constraints are present, Line 8 should be slightly modified by adding the disjunct "or $scp[c] = \{x\}$". Remark: a realistic alternative is to consider that all unary constraints are initially handled (at construction time), meaning that all unary constraints are made GAC initially, and so become entailed, which allows us to discard them definitively.

12. Page 264, Figure 5.12. The arc labelled with 0,1 leaving the leftmost state should be suppressed.

13. Page 289, Algorithm 46. Note that instead of cycling through everything until a round is finished without modification, one could certainly use some timestamps and stop early (when we are sure of having iterated over all values without any more removal).

14. Page 314, Table 6.1. Although, on page 128 we said that Golomb rules will be called "ruler", we unfortunately used "gr" instead of "ruler" in Table 6.1.

15. Page 314, Table 6.2. Considering queens, times given for FC and MAC should be considered with some caution. Currently, both FC and MAC (in my solver ACE) can easily solve the specified instances.

16. Page 407, Table 9.1. Names of some heuristics must be changed: ddeg/dom $\rightarrow$ dom/ddeg ; wdeg/dom $\rightarrow$ dom/wdeg

17. Page 426, Line -5. We indicate that "values will be assigned first to x_i and then x_j". This is rather arbitrary, and we didn't check the importance of that order compared to the possible reverse one.

18. Section 10.2. Although is it is not indicated in the book, with a proper mechanism learning nogoods from restarts, one can exhibit all solutions of a problem instance while restarting and without finding twice the same solution. So, ideally, you have to stop just after a negative decision (when considering binary branching). This way, you are sure of recording the relevant nogoods (guaranteeing of not finding twice the same solution).

Recent updates of this file can be found at: `www.cril.fr/~lecoutre/`