Bound Consistency on Linear Constraints in Finite Domain Constraint Programming

Yuanlin Zhang and **Hui Wu** [1]

1 Introduction

This paper discusses the complexity of bound consistency on n-ary linear constraint system and investigates the relationship between equivalent binary equation systems from the perspective of bound consistency techniques. We propose an efficient bound consistency enforcing algorithm whose complexity is $\mathcal{O}(n^2ed)$.In addition, by transforming a binary equation system into solved form, an efficient consistency enforcing algorithm can be achieved.

2 Bound Consistency on an n_ary Linear Constraint System

n_ary linear constraints are a powerful modeling tool in finite domain constraint programming. For solving such constraints, more specialized techniques rather than general CSP techniques [11] are used. One idea is to relax the traditional consistency to bound propagation [4], which is called bound consistency in this paper. [1] and [7] use the same idea to deal with general mathematical constraint. We give a formal analysis of bound consistency on n_ary linear constraints below.

Definition 1 A linear constraint cs_j *is*

$$a_{j_1}x_{j_1} + a_{j_2}x_{j_2} + \cdots + a_{j_n}x_{j_n} \diamond b_j$$
$$x_{j_i}, a_{j_i}, b_j \in Z \quad \diamond \in \{=, \leq, \geq\}.$$

We use *vars*(cs_j) and $|cs_j|$ to denote respectively the set and the number of variables that occur in cs_j. Due to lack of space, we assume that $\diamond$ is $=$, but the other cases are similar.

Definition 2 *A* linear constraint system *is a triple* (V, D, C) *where* V *denotes a set of variables* $\{x_1, x_2, \cdots, x_m\}$*,* D *denotes a set of domains* $\{D_1, D_2, \cdots, D_m\}$ *,*D_i *being the finite integer domain of* x_i*, and* C *denotes a set of constraint* $\{cs_1, cs_2, \cdots, cs_e\}$*,*$cs_i$ *being linear constraint.*

Hereafter, m,e, n and d refer to the number of variables,number of constraints, $max\{|cs_i| \mid cs_i \in C\}$, and $max\{|D_i| \mid D_i \in D\}$ respectively.

Definition 3 *A* Z-interval *is*

$$[a, b] = \{r \in R | a \leq r \leq b, a, b \in Z\}.$$

Let $\mathcal{Z}$ be the set of all Z-intervals on which $\leq$ is defined as $\subseteq$. The arithmetic operations on Z-interval is as those in interval arithmetic [8].

[1] Department of Information Systems and Computer Science, National University of Singapore,Singapore.

ECAI 98. *13th European Conference on Artificial Intelligence*
Young Researcher Paper
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

Definition 4 *The* Z-interval representation *of a set* $S \in R$ *is*

$$\Box S = max\{[a, b] \in \mathcal{Z} | [a, b] \subseteq S\}.$$

In the following presentation, $[x_i]$ is used to represent the Z-interval associated with x_i, and $\langle x_i \rangle$ is used to indicate the vector of lower and upper bounds of variable x_i.

Definition 5 *The* projection function π_i *of a constraint* cs_j *on* x_i *is*

$$\pi_i(cs_j) = \frac{-1}{a_i}(a_1x_1 + \cdots + a_{i-1}x_{i-1} + a_{i+1}x_{i+1} + \cdots + a_nx_n - b_j).$$

Let

$$\Pi_i(cs_j) = \tfrac{-1}{a_i}[a_1[x_1] + \cdots + a_{i-1}[x_{i-1}] + a_{i+1}[x_{i+1}] + \cdots + a_n[x_n] - b_j].$$

Definition 6 *The* projection *of a constraint* cs_j *on variable* x_i *is a set*

$$\{v_i \in R \mid \exists v_k \in [x_k], k \neq i \text{ such that } cs_j(v_1, \cdots, v_n) \text{ holds }\}.$$

The projection of cs_j on variable x_i is exactly $\Pi_i(cs_j)$.

Definition 7 *A constraint* cs_j *is* bound consistent *with respect to* $([x_1], \cdots, [x_m])$ *iff* $\forall x_i \in$ vars(cs_j), $[x_i] \subseteq \Box\Pi_i(cs_j)$.
A constraint system is bound consistent *with respect to* $([x_1], \cdots, [x_m])$ *iff every* $cs_j \in C$ *is bound consistent.*

Definition 8 *An interval vector* $\tilde{x} = ([x_1], \cdots, [x_n])$*, where* $[x_i] \subseteq \Box D_i$*, is a* fixed point *of a constraint system if the constraint system is bound consistent with respect to* $\tilde{x}$*.*

Algorithm BC
begin
  $Q \leftarrow \{< x_i, cs_j > | \forall cs_j \in C, \forall x_i \in vars(cs_j)\}$;
  while(Q not empty)
  begin
    select and delete $< x_i, cs_j >$ from Q;
    if Revise($< x_i, cs_j >$)
      $Q \leftarrow Q \cup \{< x_l, cs_k > | \forall l, cs_k \;\; x_l, x_i \in vars(cs_k), l \neq i\}$
  end
end
function Revise($< x_i, cs_j >$)
begin
  if not ($[x_i] \subseteq \Box\Pi_i(cs_j)$)
    begin
      $[x_i] \leftarrow [x_i] \cap \Box\Pi_i(cs_j)$;
      return true
    end
  else return false
end

The above algorithm is a natural extension of AC-3 [9].It always achieves the greatest fixed point of the constraint system.

Proposition 1 *For constraint system* (V, D, C)*,the worst case complexity of the algorithm is* $\mathcal{O}(n^3 ed)$.

Observe that the computation of projections of constraint cs_j on the involved variables is closely related.Consider cs_j which is of the form $a_1x_1 + \cdots + a_nx_n = b_j$. Let

$$f_j(x) = a_1x_1 + a_2x_2 + \cdots + a_nx_n - b_j$$

where

$$x = (x_1, \cdots, x_n),$$

and its natural interval extension be

$$F_j([X]) = a_1[x_1] + a_2[x_2] + \cdots + a_n[x_n] - b_j$$

where

$$[X] = ([x_1], \cdots, [x_n]).$$

Now,we have

$$\Pi_i(cs_j) = -\frac{1}{a_i}[\langle F_j([X])\rangle - \langle a_i[x_i]\rangle].$$

By introducing an auxiliary variable y_j for cs_j and x_{ji} for variable x_i, the function Revise() can be implemented in constant time. Algorithm BC can be modified as follows. At the beginning of the algorithm, evaluate $[y_j] = F_j(\Box D_1, \cdots, \Box D_m)$ and $[x_{ji}] = \Box D_i$, $[x_i] = \Box D_i$. When function Revise($< x_i, cs_j >$) returns true, we evaluate related $[y_k]$s

$$\forall k \;\; x_i \in vars(cs_k)$$
$$[y_k] \leftarrow [\langle y_k\rangle - \langle a_i[x_{ki}]\rangle] + a_i[x_i], \;\; [x_{ki}] \leftarrow [x_i].$$

In the Revise() function, let

$$\Pi_i(cs_j) = -\frac{1}{a_i}[\langle y_j\rangle - \langle a_i[x_{ji}]\rangle].$$

Proposition 2 *For system* (V, D, C)*, the complexity of the modified algorithm is* $\mathcal{O}(n^2 ed)$.

An n-ary constraint(with $n > 3$) can be transformed to a number of ternary constraints by introducing intermediate variables which will be involved in the propagation [2]. The transformed constraint system has following property.

Proposition 3 *Bound consistency can be enforced on the transformed constraint system by Algorithm BC in* $\mathcal{O}(ned')$ *where*

$$d' = \max_j \min\left\{\sum_{i=1}^{(l-2)/2} |a_{ji}|d, \;\; \sum_{i=l/2}^{l-2} |a_{ji}|d\right\}$$
$$l = |vars(cs_j)|.$$

Note that d' is related to n and the magnitude of the coefficients in C.

3 Binary equation system

Binary linear constraint system is an important case in finite domain constraint programming [3]. Efforts such as [5] have been made to improve the efficiency of consistency enforcing algorithm on binary system. Here, we focus on binary equation system.

A binary equation system may have many equivalent systems which have the same solution set. It is desirable to find a system on which an efficient consistency algorithm can be constructed.

A binary equation system is in solved form if C is of the form $bx_B = ax_N + c$ where $x_B \cup x_N = V$ and $x_B \cap x_N = \emptyset$.

For a binary equation system (V, D, C), we have following bound consistency algorithm

- Transforming C into its solved form by Gaussian-Jordan elimination,
- Selecting a special order to revise the bounds of variables.

Proposition 4 *The worst case complexity of the above algorithm is* $\mathcal{O}(e \log a)$*, where* a *is the greatest coefficient ever produced in Gaussian-Jordan elimination.*

Note that the complexity of the above algorithm does not depend on the size of domain any longer.

ACKNOWLEDGEMENTS

The authors give special thanks to Joxan Jaffar and Roland Yap for invaluable advice and many discussions.

REFERENCES

[1] Benhamou,F. and Older, W., Applying Interval Arithmetic to Real, Integer and Boolean Constraints,*Journal of Logic Programming 32(1)*,1997

[2] Codognet,P. and Diaz,D., Compiling Constraints in $CLP(FD)$, *Journal of Logic Programming*,27(3), 185-226, 1996.

[3] Dincbas, M., van Hentenryck,P., Simonis,H. and Aggoun, A., The Constraint Logic Programming Language CHIP, *Proceedings of the 2nd International Conference on Fifth Generation Computer Systems*, 249-264, 1988

[4] van Hentenryk, P., *Constraint Satisfaction in Logic Programming*, MIT Press, Cambridge, 1989

[5] van Hentenryck, P., Deville, Y. and Teng, C.-M., A Generic Arc-Consistency Algorithm and its Specializations, *Artificial Intelligence 58*, 291-321, 1992

[6] Jaffar, J. and Maher, M. J., Constraint Logic Programming,*Journal of Logic Programming* 19/20, 503-581,1994

[7] Lhomme, O., Consistency Techniques for Numeric CSPs, *Proceedings of IJCAI-93*,Chambery,France,232-238, 1993

[8] Moore, R.E., *Interval Analysis*,Prentice Hall, 1966

[9] Mackworth,A. K., Consistency in Networks of Relations, *Artificial Intelligence 8(1)*,118-126,1977

[10] Mackworth,A. K. and Freud,E.C.,The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems, *Artificial Intelligence 25*,65-74,1985

[11] Mohr,R. and Masini, G., Good Old Discrete Relaxation, *Proceedings of ECAI-88*,Munish,Germany,1988.