Chapter 5: Interval Propagation

CONSTRAINT PROGRAMMING COURSE

M1 IDO

Introduction to Interval Propagation

Overview

- **Constraint Propagation:** The process of ensuring domain consistency relative to each constraint.
- **Interval Propagation:** A specialized case of the Constraint Satisfaction Problem (CSP) where variables represent an **infinite set of values**.
- **Domain:** Defined over real numbers R.
- **Complexity:** Often involves non-linear and non-polynomial equations (e.g., $\sin(x) + e^y \leq 10$), where traditional symbolic algebra often fails.

Motivation for Real-Variable Constraints

1. Physical Systems & Engineering

- Real-world data is often **imprecise** (sensor noise, tolerances).
- Leads to **under-constrained problems** (infinite solutions).
- Requires robust handling of non-linear constraints.

2. Practical Example: Ohm's Law

- Equation: $U = R \cdot I$
- Suppose we know: $10.5 \leq R \leq 11$ and $1 \leq I \leq 2$.
- Propagation allows us to deduce the bounds for U: $10.5 \leq U \leq 22$.

N-ary Constraints and Consistency Levels

Handling multiple variables

- How do we manage constraints involving more than two variables?
- **Hyper-arc Consistency:** An extension of arc consistency for n variables.
- **The Challenge:** Determining hyper-arc consistency is computationally expensive and mathematically complex for non-linear functions.
- **The Solution: Bounds Consistency (2B-Consistency)**. We focus solely on the lower and upper limits of the intervals.

Defining Bounds Consistency (2B-Consistency)

- In an arithmetic CSP, we represent the domain of x as an interval [l, u], representing all values $\{v \in \mathrm{R} \mid l \leq v \leq u\}$.
- **min(D, x) and max(D, x):** The infimum and supremum of the domain.
- **Definition:** A constraint C is **2B-Consistent** for variable x if:
  - When x = min(D, x), there exist values for all other variables within their respective bounds that satisfy C.
  - When x = max(D, x), there exist values for all other variables within their respective bounds that satisfy C.

Arc Consistency vs. Bounds Consistency

A Comparison of Strength

- **Arc Consistency (AC):** Stronger. Every single value in the domain must have a "support" in the other variables' domains.
- **Bounds Consistency (BC):** Weaker. Only the boundaries (min/max) are guaranteed to have support.

Example of the Gap:

- Problem: $x = y^2$ with $Dx = [1, 4]$ and $Dy = [-2, 2]$.
- This is **Bounds Consistent** because for x=1, y=1 works; for y=2$ x=4 works.
- However, it is **not Arc Consistent** because y=0 is in Dy, but there is no $x \in [1,4]$ such that $0^2 = x$.

Mechanism: Interval Arithmetic Rules

To reach Bounds Consistency, we apply **narrowing rules** based on interval arithmetic:

Let $\odot$ be an operator {+, -, *, /} and intervals [a, b], [c, d].

- **Addition $\oplus$:** $[a, b] \oplus [c, d] = [a+c, b+d]$
- **Subtraction $\ominus$:** $[a, b] \ominus [c, d] = [a-d, b-c]$
- **Multiplication $\otimes$:** $[a, b] \otimes [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$
- **Division $\oslash$:** $[a, b] \oslash [c, d] = [a, b] \times [1/d, 1/c]$ (provided $0 \notin [c,d]$)

Propagation Example (Addition)

Constraint: X = Y + Z

To maintain consistency, we must satisfy three structural viewpoints:

1.X = Y + Z

2.Y = X - Z

3.Z = X - Y

Derived Propagation Rules:

- $X \in [\min(Y)+\min(Z), \max(Y)+\max(Z)]$
- $Y \in [\min(X)-\max(Z), \max(X)-\min(Z)]$
- $Z \in [\min(X)-\max(Y), \max(X)-\min(Y)]$

Step-by-Step Domain Reduction

Scenario: X = Y + Z

Initial Domains: D(X) = [4, 8], D(Y) = [0, 3], D(Z) = [2, 2]

Applying Rules:

Xnew = [0+2, 3+2] = [2, 5] $\rightarrow$ Intersection with [4, 8] yields **[4, 5]**.

Ynew = [4-2, 8-2] = [2, 6] $\rightarrow$ Intersection with [0, 3] yields **[2, 3]**.

Znew = [4-3, 8-0] = [1, 8] $\rightarrow$ Intersection with [2, 2] yields **[2, 2]**.

Final Filtered Domains: D(X)=[4, 5], D(Y)=[2, 3], D(Z)=[2, 2].

Handling Inequalities and Disequalities

1. Inequalities (e.g., $Y \leq Z$):

Very effective for pruning. If max(Y) > max(Z) then max(Y) is reduced to max(Z).

2. Disequalities (e.g., $Y \neq Z$):

Propagation is generally **weak**.

- If D(Y)=[2, 4] and D(Z)=[2, 3], no pruning occurs because multiple values still satisfy the constraint.
- **Pruning only occurs when one variable is assigned a fixed value** that matches the boundary of the other.
- *Example:* D(Y)=[2, 4], D(Z)=[2, 2] $\rightarrow$ D(Y) becomes [3, 4].

Complex Propagation: Multiplication

Constraint: $X = Y \times Z$

- **If all variables are positive:** Propagation is straightforward multiplication of bounds.
  - $X \geq \min(D, Y) * \min(D, Z)$, $X \leq \max(D, Y) * \max(D, Z)$ etc. for Y , Z
- **General Case:** We must consider all combinations of signs.
  - Xmin = min(min(Y)min(Z), min(Y)max(Z), max(Y)min(Z), max(Y)max(Z))
- **Division Caution:** When propagating to Y or Z (e.g., Y = X/Z), if $0 \in D(Z)$, the domain of Y remains unrestricted until the sign of Z is strictly determined (non-negative or non-positive).

The Bounds Consistency Algorithm

BoundsCons(C, D):

- Apply propagation rules for every constraint in set C.
- Repeat until a **Fixed Point** is reached (no further domain changes).
- **Optimization:** Only re-examine constraints whose variables have been updated.

Solver Integration:

- If D becomes empty Return **Infeasible**.
- If all variables are instantiated Return **Satisfiable**.
- Else Return **Unknown** (Search required).

Search: Backtracking with Bounds Consistancy (BC)

Standard solving involves interleaving **Filtering** and **Searching**:

1. Apply **Bounds Consistency** to prune the search space.
2. Select a variable and branch (assign a value or split the interval).
3. Recursively apply BC after each assignment.

Benefit: This significantly reduces the depth of the backtracking tree by detecting failures early.

Synthesis: The Knapsack Problem

Constraints:

$4J + 3P + 2C \leq 9$ (Weight Capacity)

$15J + 10P + 7C \leq 30$ (Minimum Profit)

Initial Domains: D(J, P, C) = [0, 9]

After BC on Constraint 1: D(J)=[0, 2], D(P)=[0, 3], D(C)=[0, 4].

Branching on J=0: BC deduces D(P)=[1, 3], D(C)=[0, 3].

Branching on P=1: BC deduces D(C)=[3, 3].

Solution Found: {J=0, P=1, C=3}.

GNU Prolog Implementation

The logic we discussed is implemented in GNU Prolog using the # operator for Finite Domain constraints.
Prolog

```
% Knapsack Problem Implementation
knapsack([J, P, C]) :-
    fd_domain([J, P, C], 0, 9),
    4*J + 3*P + 2*C #=< 9,
    15*J + 10*P + 7*C #>= 30,
    fd_labeling([J, P, C]).

% Sample Query Result:
% J=0, P=1, C=3;
% J=0, P=3, C=0;
% J=1, P=1, C=1;
% J=2, P=0, C=0.
```

Conclusion: By combining interval propagation and backtracking, CLP solvers efficiently navigate infinite or large discrete search spaces to find optimal solutions.