An Incremental Constraint Satisfaction Algorithm for Dynamic Reconfiguration

Sina Entekhabi
Ahmet Serkan Karataş
Halit Oğuztüzün
ODTÜ, Ankara
IZTECH Dependability, 8 May 2017

Supported by TÜBİTAK-ARDEB-1001 program under project 215E188.
Outline

• Introduction
• Problem definition
• Related works
• Incremental algorithm
• Tracing example
• Conclusion
• References
Introduction (1/4)

- **Software Product Line (SPL)**
  - A series of similar systems
  - Sharing common cores with some differences
  - Variability management before runtime
  - Ex: smartphones

- **Dynamic SPL (DSPL)**
  - Variability management at runtime
  - Ex: Smart homes
Introduction (2/4)

• Variability Management
  o Ex: Feature model (FM) diagram
  o SPL: Some of the features in a product
  o DSPL:
    • All of the features in a DSPL product
    • Runtime reconfigurations regarding context condition

Feature model diagram of a smart home [3]
Introduction (3/4)

• Constraint Logic Program
  o Containing constraints in the body of clauses
  o Ex: A( x, y):- x>0, y>1, B(x)

• FM relations can be expressed as clauses of logical expressions
  o Ex:
    • “A excludes B” as “¬(A ∧ B)”
    • “A requires B” as “A ⇒ B”
    • “A is the parent of B, in a mandatory relation” as “A ⇔ B”
    • “A is the parent of B, in an optional relation” as “B ⇒ A”
    • “A is the parent of B and C, in an ‘OR’ relation” as “B ∨ C ⇒ A”
    • “A is the parent of B and C, in an alternative relation” as “((B ∧ ¬C) ∨ (¬B ∧ C)) ⇔ A”
Introduction (4/4)

• Runtime DSPL reconfiguration

The context monitor specifies activation and/or deactivation of some of the features in specific conditions [4]

<table>
<thead>
<tr>
<th>Context Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition 1</td>
</tr>
<tr>
<td>Condition 2</td>
</tr>
<tr>
<td>....</td>
</tr>
<tr>
<td>Condition N</td>
</tr>
</tbody>
</table>

• Effective reconfiguration criteria:
  o Imposing the minimum number of changes to the current product
Problem Definition (1/2)

• The whole FM as a constraint network
  o Every relation as a constraint
  o Reaching a valid DSPL product by satisfying all of the constraints

• DSPL reconfiguration problem as Constraint Satisfaction Problem (CSP)
Problem Definition (2/2)

Having a constraint network including a set of variables $V$:

$$V = \{v_1, v_2, ..., v_n\} \text{ where } v_i \in D_i \text{ for } 1 \leq i \leq n,$$

and a set of satisfied constraints $C$ among variables in $V$:

$$C = \{c_1, c_2, ..., c_k\},$$

and a resolution $R$:

$$R = \{v_{j_1} \leftarrow a'_1, v_{j_2} \leftarrow a'_2, ..., v_{j_m} \leftarrow a'_m\},$$

where the variables have the previous values:

$$v_{j_r} = a_r \text{ for } 1 \leq r \leq m,$$

the aim is satisfying $C$ and $R$ while minimizing the condition $\theta$ below:

$$\theta = \sum_{r=1}^{m} a_r \oplus a'_r,$$

where $x \oplus y = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$
Related works

• Incremental CSP algorithms for constraint hierarchy
  o EX: DeltaBlue, SkyBlue, cassowary

• Dynamic CSP algorithms
  o The number of constraints and/or variables are variable
  o Using previous solution or learning to reach next solution
Incremental algorithm

• Our incremental algorithm is inspired from SkyBlue
• Using the concept of multi-directional methods
• The data structure includes these parts below:
  o S-Variable
  o S-Method
  o S-Constraint
  o S-network
  o S-log

• Our algorithm includes two main functions:
  o Reconfigure function
  o Solve function
Algorithm 1 Reconfigure Function

Input:
  a reconfiguration request and a consistent constraint network

Output:
  a list including variable changes which results in satisfying the request and a consistent constraint network or an empty list

1: function RECONFIGURE(V, System) ▷ V - a list of variables, System - constraint network
2:   V₀ ← System.getVariables(getNames(V))
3:   newlyChangedVars ← System.setVariables(V)
4:   log1.assignedVariables ← V
5:   log1.changedVariables ← newlyChangedVars
6:   relatedCS ← System.relatedConstraints(newlyChangedVars)
7:   newCS ← sensitiveConstraints(relatedCS, newlyChangedVars)
8:   if newCS is empty then
9:     System.setVariables(V₀)
10:    return newlyChangedVars
11:  else
12:    result ← SOLVE(newCS, log1, System, { })
13:    System.setVariables(V₀)
14:    return result
15:  end if
16: end function
Tracing example

- Supposing a DSPL with the FM diagram below.
- Request **R**: activate Feature B and C
Arbitrary reconfiguration

- An arbitrary valid reconfiguration satisfying $R$
  - changes :5
- Valid reconfigurations with less than 5 changes exists
FM to constraint network

- Corresponding FM to a set of variables and constraints among them
Constraint definitions

- **C1**: \( B \Rightarrow A \)
- **C2**: \( C \Rightarrow A \)
- **C3**: \( D \Leftrightarrow A \)
- **C4**: \( \sim (B \land E) \)
- **C5**: \( (F \land \sim G \land \sim E) \lor (\sim F \land G \land \sim E) \lor (\sim F \land \sim G \land E) \Leftrightarrow C \)
- **C6**: \( (I \lor H) \Leftrightarrow E \)
- **C7**: \( G \Leftrightarrow L \)
- **C8**: \( F \Leftrightarrow J \)
- **C9**: \( F \Leftrightarrow K \)
Different representation

- Representing constraint network similar to FM
Tracing (1/11)

- Satisfying the request R as the first step
- Distributing the effects at the next steps
Tracing (2/11)

• A is requested to be true by C1 and C2
• A was true beforehand, no more distribution from A side

Variable assigned to True
Variable assigned to False
Variable needed to be assigned to True to reach a solution
Variable needed to be assigned to False to reach a solution
Tracing (3/11)

- E is requested to be False by C4
Tracing (4/11)

- C5 needs G or F be True, but not E
- Choosing G arbitrarily at this point

![Diagram](image)
Tracing(5/11)

• C7 needs L be True.
Tracing(6/11)

- No more solution to recheck: one Solution found
- Solution 1: change (B,C,G,L) to true, changes: 4

Diagram:

- Variables assigned to True:
  - D
  - C
  - E
- Variables assigned to False:
  - B
  - F
- Variables needed to be assigned to True to reach a solution:
  - G, L
- Variables needed to be assigned to False to reach a solution:
  - A, H, I, J, K

Diagram nodes labeled with lowercase letters correspond to the variables, and edges labeled with lowercase letters indicate the conditions or constraints.
Tracing (7/11)

- Backtrack: to satisfy C5, F can be True as well.
- Choosing F and trying to find a solution
Tracing (8/11)

- C8 needs J be True.
- Having 4 changes up to now in this solution search.
Tracing(9/11)

• Having 4 changes up to now in this solution search.
• Solution1 had 4 changes as well. Pruning this branch
The algorithm return an optimum solution, solution1.
Solution1 is the only optimum solution in this case.
After applying solution 1 to the system, the FM of the system would be the diagram below.

```
Activated feature

Deactivated feature
```
Conclusion

• Variability management of DSPLs by FM
• FM corresponds to constraint logic program
• Dynamic reconfiguration in DSPLs as CSP
• Effective reconfiguration by incremental algorithms
References (1/2)

References (2/2)


Thank you!