Contribution to Geometric Constraint Solving in Cooperative Engineering

*PhD Dissertation*

Sebastià Vila-Marta

Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics
Programa de Doctorat en Software
Framework

- This work focuses on constructive geometric constraint solving.
- It builds on:
What is this dissertation about?

- The architecture of constructive geometric constraint solvers.
- Characterization of the constructive analysis methods domain.
- Conditional completion of under constrained problems.

In addition,

- Multiple views geometric constraint models.
Contributions (I)

► Architecture:
  • Software architecture for constructive solvers.
  • The index and the selector operation.
  • Definition of a geometric constraint based model.

► Domain characterization:
  • Equivalence of Fudos and Owen analysis domains.
  • Structure of constraint graphs class.
  • Recursive definition of solvable constraint graphs.
  • Tree decompositions: a useful theoretical and practical tool.
Contributions (II)

- Completion:
  - Simple solution for free completion based on decomposition trees.
  - Non optimality of greedy approach for conditional completion.
  - Good experimental behavior of greedy algorithm.

- Multiple views model:
  - Definition of a basic set of tools for constraint based models.
  - Description of a multiple views model based on these tools.
1. Architecture of constructive solvers.
2. Domain characterization of analyzers.
3. The completion operation.
4. Multiple views model.
5. Conclusions and future work.
Architecture of constructive solvers
Background

- Most of published work briefly sketch an architecture for the solvers that they propose.
- The issue has not explicitly been addressed in depth before.
- Here, an architecture is defined and new concepts like that of index and selector are introduced.
- A new version of the SolBCN solver is organized according this ideas.
Architecture diagram

- Abstract problem
- Analyzer
- Abstract plan
- Selector
- Parameter assignment
- Constructor
- Index assignment
- Geometry assignment
Architecture diagram
Architecture diagram

p1=(0,0)
p3=(d1,0)
l3=\text{line}(p1,p3)
laux=\text{parallel}\ -\ ld(l3,h1)
c=\text{circle}\ -\ cr(p1,d2)
p2=\text{intersec}\ -\ cl(c,laux,s)
l1=\text{line}(p1,p2)
l2=\text{line}(p2,p3)
Architecture diagram

p1=(0,0)
p3=(d1,0)
l3=line(p1,p3)
laux=parallel–ld(l3,h1)
c=circle–cr(p1,d2)
p2=intersec–cl(c,laux,s)
l1=line(p1,p2)
l2=line(p2,p3)
Abstract problem

Analyzer

Abstract plan

Selector

Geometry assignment

Index assignment

p1=(0,0)
p3=(d1,0)
l3=line(p1,p3)
laux=parallel ld(l3,h1)
c=circle cr(p1,d2)
p2=intersection cl(c,laux,s)
l1=line(p1,p2)
l2=line(p2,p3)
Architecture diagram

Abstract problem

Analyzer

Abstract plan

Selector

Parameter assignment

d1=6,0
d2=3,0
h1=2,5

Constructor

Index assignment

Geometry assignment

Contribution to GCS – p.9/44
Architecture diagram

Abstract problem
Analyzer
Abstract plan
Selector
Parameter assignment
Index assignment

s=−1  s=+1
Architecture diagram

Abstract problem

Analyzer

Abstract plan

Selector

Parameter assignment

p1=(0,0)
p2=(1.7,2.5)
p3=(6,0)
l1=[y=1.5x]
l2=[y=−0.6x+3.5]
l3=[y=0]

Constructor

Geometry assignment

Index assignment
Index assignment concept

- An **index assignment** is an assignment to the set of “specific parameters” that select the solution in geometric intersections.

- The index assignment identifies a particular point in the solutions space of an abstract problem, and:
  - The index is local to an abstract plan.
  - Particular combinations of index and parameter assignments can be meaningless.
  - Distinct index assignments can lead to the same realization (modulo rigid transformations).
Constraint based geometric model

\[ s_1 = -1 \]
\[ s_2 = +1 \]

\[ d_1 = 1 \]
\[ d_2 = 2.6 \]
\[ h_1 = 2.2 \]
Publications

Most of this work has been published in:

Domain characterization of analyzers
Distinct constructive analysis algorithms have been published by Owen, Fudos, Todd and other authors.

Some of them are based on geometric constraint graphs analysis.

Here, we proof that the domain of Owen and Fudos methods is the same by using tree decompositions.

This domain is characterized by a recursive definition.
Tree decomposition of a graph

\{a, b, c, d, e, f\}
Tree decomposition of a graph
Tree decomposition of a graph

\{a, b, c, d, e, f\}
\{b, c\} \{a, c, d, e\} \{a, b, f\}
\{a, b\} \{f, b\} \{f, a\}
Tree decomposition of a graph

```
{a,b,c,d,e,f}
  /   \
{c,d}  {c,e}  {a,d,e}
  |      |      |
{a,d}  {d,e}  {a,e}
  |      |      |
{a,b}  {f,b}  {f,a}
```

```
{a,b,c}  {a,c,d,e}  {a,b,f}
  /   \
{b,c}  {a,c,d,e}  {a,b,f}
  /   \
{c,d}  {a,d,e}  {a,b}
  /   \
{a,d}  {a,e}  {a,b}
  /   \
{a,d}  {d,e}  {a,e}
  /   \
{a,d}  {d,e}  {a,e}
```
Domain equivalence of constructive methods

Let $G = (V, E)$ be a well constrained geometric constraint graph. The following assertions are equivalent:

1. $G$ is full tree decomposable.
2. $G$ is s-tree decomposable (Owen).
3. $G$ is solvable by reduction analysis (Fudos).
4. $G$ is solvable by decomposition analysis (Fudos).

The class of graphs fulfilling the above properties is named the constructively solvable graphs class.
Generation of solvable graphs

The solvable graphs class $\mathcal{G}$ is recursively defined:

- The triangle belongs to the class $\mathcal{G}$
- Let $G_1, G_2 \in \mathcal{G}$, obtain $G_3$ by replacing an edge of $G_2$ by $G_1$, then $G_3 \in \mathcal{G}$.
- No other graph belongs to the class.
The solvable graphs class $\mathcal{G}$ is recursively defined:

- The triangle belongs to the class $\mathcal{G}$
- Let $G_1, G_2 \in \mathcal{G}$, obtain $G_3$ by replacing an edge of $G_2$ by $G_1$, then $G_3 \in \mathcal{G}$.
- No other graph belongs to the class.
The solvable graphs class $\mathcal{G}$ is recursively defined:

- The triangle belongs to the class $\mathcal{G}$
- Let $G_1, G_2 \in \mathcal{G}$, obtain $G_3$ by replacing an edge of $G_2$ by $G_1$, then $G_3 \in \mathcal{G}$.
- No other graph belongs to the class.
The solvable graphs class $\mathcal{G}$ is recursively defined:

- The triangle belongs to the class $\mathcal{G}$
- Let $G_1, G_2 \in \mathcal{G}$, obtain $G_3$ by replacing an edge of $G_2$ by $G_1$, then $G_3 \in \mathcal{G}$.
- No other graph belongs to the class.
The solvable graphs class $\mathcal{G}$ is recursively defined:

1. The triangle belongs to the class $\mathcal{G}$
2. Let $G_1, G_2 \in \mathcal{G}$, obtain $G_3$ by replacing an edge of $G_2$ by $G_1$, then $G_3 \in \mathcal{G}$.
3. No other graph belongs to the class.
Generation of solvable graphs

The solvable graphs class $\mathcal{G}$ is recursively defined:

- The triangle belongs to the class $\mathcal{G}$
- Let $G_1, G_2 \in \mathcal{G}$, obtain $G_3$ by replacing an edge of $G_2$ by $G_1$, then $G_3 \in \mathcal{G}$.
- No other graph belongs to the class.
Publications (I)

Most of this work has been published in:

Most of this work has been published in:


The completion operation
To complete an under constrained abstract problem means to add new constraints until it becomes well constrained.

Additional constraints can be:

- Taken from a given set: conditional completion.
- Taken from the universal set: free completion.

Here, a method based on tree decomposition is presented.
Free completion
Free completion
Free completion
Structure of geometric constraint graphs class

$G$

Over-constrained  Well-constrained  Under-constrained

Tree decomposable
Solvable  Completable
Conditional completion of $G = (V, E)$ requires to take the maximum number of edges from an additional set $\hat{G} = (V, \hat{E})$. 
Conditional completion

Conditional completion of $G = (V, E)$ requires to take the maximum number of edges from an additional set $\hat{G} = (V, \hat{E})$. 
Conditional completion leads to a combinatorial optimization problem defined by the subset system $(\mathcal{E}, \mathcal{S})$ and the weight function $w$ as follows:

- $\mathcal{E}$: the set of edges $E \cup \hat{E}$
- $\mathcal{S}$: the tree decomposable subsets of $\mathcal{E}$
- $w$: $w(e) = 2$ if $e \in E$; $w(e) = 1$ if $e \in \hat{E}$
**Greedy algorithm**

```plaintext
algorithm GreedyOptimize
    \[ I := \emptyset \]
    \[ \textbf{while} \ \mathcal{E} \neq \emptyset \ \textbf{do} \]
    \[ \quad e := \text{a maximum weight member of } \mathcal{E} \]
    \[ \quad \mathcal{E} := \mathcal{E} \setminus \{e\} \]
    \[ \textbf{if} \ I \cup \{e\} \in S \ \textbf{then} \ I := I \cup \{e\} \]
```
Greedy algorithm example
Greedy algorithm example
Greedy algorithm example
Greedy algorithm example

Contribution to GCS – p.27/44
Greedy algorithm example
Greedy algorithm example

Non-optimal case

```
Non-optimal case
```

Contribution to GCS – p.27/44
Greedy algorithm results

- Greedy algorithm is not optimal: in some cases it does not compute a maximal completion.
- Nonetheless, exhaustive testing shows that greedy algorithm fails in approximately 3.5% of the cases.
- If we relax the need for a solvable completion, then well constrained conditional completion can be solved by greedy algorithm.
Conditional completion applications

- Over constrained problems.
- Constraints with priorities.
- Work with views.
Publications (I)

Most of this works has been published in:


Most of this works has been published in:

Multiple views model
Multiple view models are important in cooperative design environments.

When design software relies on constraint based models, specific operations are needed.

Here, a set of operations that allows to work with such models is proposed.
Constraint-based models with views

- Constraint-based models $A$ and $B$ are distinct views of the same object if
  - The geometric elements and topological constraints of $A$ and $B$ are the same
  - The parameters of $A$ and $B$ are different
Repertoire of operations

- **OpenView** creates a new view $B$ from an existing one $A$ by replicating over $B$ the geometric elements and the topology of $A$.

- **Complete** a view $B$ means to add constraints to $B$ until it becomes well constrained.

- **ValuesTransfer** from view $A$ to view $B$ means to obtain the parameter values and the index value for view $B$ such that the shape of both $A$ and $B$ is the same.
Values Transfer implementation

To transfer values from $A$ to $B$:

- Measure in $A$ the parameter $p_i \in B$.
- Search the solution space of $B$ for a shape matching $A$. 

---

<table>
<thead>
<tr>
<th>GCS</th>
<th>Param</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Ap</td>
<td>$A_i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GCS</th>
<th>Param</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B$</td>
<td>Bp</td>
<td>$B_i$</td>
</tr>
</tbody>
</table>
A cooperative environment

- Assume a cooperative environment like that defined by the CVS software system (Berliner)
Illustrative work sequence
Publications

Most of this work has been published in:

Conclusions and future work
Conclusions (I)

- An architecture suitable for geometric constraint solvers has been defined. Functional units and involved data types have been identified. The selector functional unit and the index assignment have been proposed.

- We have studied the domain of three constructive analyzers. We have proved that, for well constrained problems, these domains are equivalent.
Conclusions (II)

- Owen’s algorithm has been reinterpreted and a new version which is easier to understand has been proposed.
- Tree decompositions have been defined and extensively used in the work. Tree decompositions have been of practical interest in building the new SolBCN.
- The class of constructively solvable graphs has been defined and a constructive characterization of this class has been given.
Conclusions (III)

- Completion of under constrained abstract problems have been studied. We have defined free and conditional completion.
- The conditional completion have been reduced to a combinatorial optimization problem. A greedy algorithm has been successfully applied.
- A repertoire of operations has been defined that allows to work with multiple views geometric constraint models.
Future work

- Efficient algorithm to compute tree decompositions following Ramachandran’s work on graph triconnectivity.

- Efficient algorithm to compute incremental tree decompositions after edge insertion or edge deletion.

- Prove or disprove if conditional completion is NP-hard.