Interpreting Line Drawings: An alternative introduction to Constraint Satisfaction (parallel to AIMA 6.1,6.2)

Mitch Marcus

We Interpret Line Drawings As 3D
- We have strong intuitions about line drawings of simple geometric figures:
  - We naturally interpret 2D line drawings as planar representations of 3D objects.
  - We interpret each line as being either a convex, concave or occluding edge in the actual object.

Possible Vs. Impossible Line Drawings
- We can reject some line drawings as impossible 3D objects, but only after some confusion.

"The Devil's Trident"

Interpreting Trihedral Scenes
- We will consider scenes made up only of opaque trihedral objects, objects where
  - All object faces are planar
  - Exactly three faces intersect at each vertex

Convexity Labeling Conventions
Each edge in an image can be interpreted to be either a convex edge, a concave edge or an occluding edge:
- + labels a convex edge (angled toward the viewer);
- - labels a concave edge (angled away from the viewer);
- \( \rightarrow \) labels an occluding edge. To its right is the body for which the arrow line provides an edge. On its left is space.
Huffman/Clowes Junction Labels

- A trihedral image can be automatically interpreted given only information about each junction in the image.
- Each interpretation gives convexity information for each junction.
- This interpretation is based on the junction type. (All junctions involve at most three lines.)

Arrow Junctions have 3 interpretations

The image of the same vertex from a different point of view gives a different junction type
(from Winston, Intro to Artificial Intelligence)

L Junctions have 6 interpretations

The Same “Arrow” and “L” Junctions in a Standardized Orientation

“T” Junctions Have 4 Interpretations, “Y” Junctions Have 3

Summary

<table>
<thead>
<tr>
<th>Type of Junction</th>
<th>Number of Physically Possible Interpretations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrow Junctions</td>
<td>3</td>
</tr>
<tr>
<td>L Junctions</td>
<td>6</td>
</tr>
<tr>
<td>T Junctions</td>
<td>4</td>
</tr>
<tr>
<td>Y Junctions</td>
<td>3</td>
</tr>
</tbody>
</table>
Huffman/Clowes
Line Drawing Interpretation Problem

- **Given:** a line drawing of a simple "blocks world" physical image
- **Compute:** a set of junction labels that yields a consistent physical interpretation

For Notes: The General-Viewpoint Assumption:

- H/C Interpretation assumes the General Viewpoint Assumption.
- Shifts in the position of the viewer do not affect the configuration of the line drawing.
- This rules out the possibility of the accidental alignment of image features into a spurious junction.

(drawings from Machine Interpretation of Line Drawings, Kokichi Sugihara, MIT Press)

Towards Arc Consistency:
The Edge Consistency Constraint

The Edge Consistency Constraint

Any consistent assignment of labels to the junctions in a picture must assign the same line label to any given line.

An Example of Edge Consistency

- Consider an arrow junction with an L junction to the right:
  - **A1 and either L1 or L6 are compatible** since they both associate the same kind of arrow with the line.
  - **A1 and L2 are incompatible**, since the arrows are pointed in the opposing directions.
  - **Similarly, A1 and L3 are incompatible.**

An Example of Edge Consistency II

- **A2 is compatible with ____?**
- **A3 is compatible with ____?**

(Ass: (A2, L4) (A3, L5))
A First Attempt:
The Generate and Test Algorithm

1. **Generate** all conceivable labelings.
2. **Test** each labeling to see whether it violates the edge consistency constraint; throw out those that do.

**Complexity:**
- Each junction has on average 4.5 labeling interpretations.
- If a figure has \( N \) junctions, we would have to generate and test \( 4.5^N \) different labelings.
- Thus the Generate and Test Algorithm is \( O(4.5^N) \).

Generate & Test: A Useful Algorithm??

- A picture with 27 junctions, like the devil’s trident) will not be rejected until all \( 4.5^{27} \) hypotheses have been rejected, leaving no interpretation.
  \[ 4.5^{27} = 43324930221073824.244664378464222 \]
- A computer capable of checking for edge consistency at a rate of 1 hypothesis per nanosecond would take about 1,000 years to establish that the devil’s trident has no consistent interpretation!

Improving on Generate & Test

- The Generate and Test algorithm considers too many hypotheses that are simply outright impossible according to the edge consistency constraint.
- The Solution:

Locally Consistent Search

- We can exploit locally consistent labelings to construct a globally consistent interpretation:
  1. Only add a junction label if it is consistent with its immediate neighbors;
  2. If an interpretation cannot be continued because no consistent junction label can be added, abandon that interpretation immediately.

Implementing edge consistency using search trees

1. Select some junction as the root.
2. Label children at the 1st level with all possible interpretations of that junction.
3. Label their children with possible consistent interpretations of some junction adjacent to that junction.
4. Each level of the tree adds one more labeled node to the growing interpretation.
5. Leaves represent either futile interpretations that cannot be continued or full interpretations of the line drawing.
The Fourth Level of the Search Tree

Another Idea...

We are doing somewhat better than the Generate and Test algorithm by not examining useless paths. Nevertheless, we are not fully exploiting the structure of the problem:

- Consider a picture with disconnected “islands” (junctions which are not connected by a path along any series of edges).
- All our current search methods will explore and re-explore these islands even though they do not interact with the rest of the picture!! This is very wasteful.....

Constraint Propagation Invented...

Dave Waltz’s insight:
- Pairs of adjacent junctions (junctions connected by a line) constrain each other’s interpretations!
- These constraints can be propagated along the connected edges of the graph.

Waltz Filtering:
- Start with each junction labeled with all possible interpretations.
- Suppose junctions i and j are connected by an edge. Remove any labeling from i that is inconsistent with every labeling in the set assigned in j.
- By iterating over the graph, the constraints will propagate.

When to Iterate, When to Stop?

The crucial principle:
Any algorithm must ensure that:

If a label is removed from a junction i, then the labels on all of i’s neighbors will be reexamined.

Why? Removing a label from a junction may result in one of the neighbors becoming edge inconsistent, so we need to check...

The Waltz/Mackworth Constraint Propagation Algorithm

1. Associate with each junction in the picture a set of all Huffman/Clowes junction labels appropriate for that junction type;
2. Repeat until there is no change in the set of labels associate with any junction:
   3. For each junction i in the picture:
      4. For each neighboring junction j in the picture:
         5. Remove any junction label from i for which there is no edge-consistent junction label on j.

An Example of Constraint Propagation
Inefficiencies: Towards AC-3

1. At each iteration, we only need to examine junctions \( i \) where at least one neighbor \( j \) has lost a label in the previous iteration.

2. If \( j \) loses a label only because of edge inconsistencies with \( i \), we don’t need to check \( i \) on the next iteration.

3. The only way in which \( i \) can be made edge-inconsistent by the removal of a label on \( j \) is with respect to \( j \) itself. Thus, we only need to check that the labels on the pair \((i,j)\) are still consistent. These insights lead to a new representation and a much better algorithm...

Arcs (We’re back into the book at 6.2.2)

Given this notion of \((i,j)\) junction pairs,

- The crucial notion is an arc, where an arc is a directed pair of neighboring junctions.
- For a neighboring pair of junctions \( i \) and \( j \), there are two arcs that connect them: \((i,j)\) and \((j,i)\).

Arc Consistency

We will define the algorithm for line labeling with respect to arc consistency as opposed to edge consistency:

- An arc \((i,j)\) is arc consistent if and only if every label on junction \( i \) is consistent with some label on junction \( j \).

- To make an arc \((i,j)\) arc consistent, for each label \( l \) on junction \( i \):
  - if there is no label on \( j \) consistent with \( l \) then remove \( l \) from \( i \).

(Any consistent assignment of labels to the junctions in a picture must assign the same line label to an arc \((i,j)\) as to the inverse arc \((j,i)\).)

The AC-3 Algorithm (for Line Labeling)

1. Associate with each junction in the picture a set of all Huffman/Clowes junction labels for that junction type;

2. Make a queue of all the arcs \((i,j)\) in the picture;

3. Repeat until the queue is empty:
   a. Remove an arc \((i,j)\) from the queue;
   b. For each label \( l \) on junction \( i \), if there is no label on \( j \) consistent with \( l \) then remove \( l \) from \( i \);
   c. If any label is removed from \( i \) then for all arcs \((k,i)\) in the picture except \((j,i)\), put \((k,i)\) on the queue if it isn’t already there.

Example of AC-3: Iteration 1

Queue: \((1,2)(2,1)(2,3)(3,2)\)
\((3,4)(4,3)(4,1)(1,4)(1,3)(3,1)\)

a. Removing \((1,2)\).

b. All labels on 1 consistent with 2, so no labels removed from 1.

c. No arcs added.

(derived from old CSE 140 slides by Mark Steedman)

AC-3: Iteration 2

Queue: \((2,1)(2,3)(3,2)(3,4)(4,3)\)
\((4,1)(1,4)(1,3)(3,1)\)

a. Removing \((2,1)\).

b. L2 & L4 on 2 inconsistent with 1, so they are removed.

c. All arcs \((k,2)\) except for \((1,2)\) considered to add, but already in queue, so no arcs added.
AC-3: Iteration 3
Queue: (2,3)(3,2)(3,4)(4,3)(4,1)(1,4) (1,3)(3,1)
   a. Removing (2,3).
   b. L3 on 2 inconsistent with 3, so it is removed.
   c. Of arcs (k,2), (1,2) is not on queue, so it is added.

AC-3: Iteration 4
Queue: (3,2)(3,4)(4,1)(1,4)(1,3)(1,2)
   a. Removing (3,2).
   b. A2 on 3 inconsistent with 2, so it is removed.
   c. Of arcs (k,3) excluding (2,3) are on queue, so nothing added to queue.

AC-3: Iteration 5
Queue: (3,4)(4,1)(1,3)(3,1)
   a. Removing (3,4).
   b. All labels on 3 consistent with 4, so no labels removed from 3.
   c. No arcs added.

AC-3: After Remaining Iterations
Queue: empty

AC-3: Worst Case Complexity Analysis
- All junctions can be connected to every other junction,
  - so each of n junctions must be compared against n-1 other junctions,
  - so total # of arcs is n*(n-1), i.e. O(n^2)
- If there are d labels, checking arc (i,j) takes O(d^2) time
- Each arc (i,j) can only be inserted into the queue d times
- Worst case complexity: O(n^2d^3)

For planar pictures, where lines can never cross, the number of arcs can only be linear in N, so for our pictures, the time complexity is only O(nd^2)