Outline

- Review of Diagnosis and Mode Estimation
- Generalizing Mode Estimation to Optimal CSPs
  - Model-based Programming w/o State
  - Mode Estimation as Trajectory Tracking
Model-based Diagnosis

- A failure is a discrepancy between the model and observations of an artifact.
- A diagnosis restores consistency.
  1. Enumerate candidates in order of likelihood.
  2. Test novel failures by suspending constraints and testing consistency.
  4. Conflicts prune remaining candidates.

Consistency-based Diagnosis

And(i):
- $G(i)$:
  $Out(i) = In1(i) \text{ AND } In2(i)$
- $U(i)$:

  • ALL components have “unknown mode” $U$.
  • $U$ never assigned in $C$

Diagram: Diagnosis $= \{A1=G, A2=U, O1=G, O2=U, O3=G\}$

- Model: $\Psi(X,Y)$ over system variables $X$ and mode variables $Y$
- Obs: Assignment to $O \subseteq X \cup Y$
- Candidate $C_i$: Assignment of modes to $Y$
- Diagnosis $D_i$: A candidate such that $D_i \wedge \text{Obs} \wedge \Psi(X,Y)$ is satisfiable.
**Compact Encoding: Kernel Diagnoses**

Kernel Diagnosis

\[ \{A2=U, M2=U\} \]

Partial Diagnosis: A set of component modes \( P_i \) all of whose extensions are diagnoses.

- \( P_i \) removes all symptoms
- \( P_i \) entails \( \Psi(X,Y) \land \text{Obs} \) (implicant)

Kernel Diagnosis: A minimal partial diagnosis \( K_i \)

- \( K_i \) is a prime implicant of \( \Psi(X,Y) \land \text{Obs} \)

**Diagnoses Found by Mapping Conflicts to Kernels**

Conflict: A set of component modes \( C_i \) that are inconsistent with the model and observations.

- \( \Psi(X,Y) \land \text{Obs} \) entails not \( C_i \)

Kernel Diagnosis: A minimal set of component modes \( K_i \) that eliminate all symptoms.

- \( K_i \) is a prime implicant of \( \Psi(X,Y) \land \text{Obs} \)
- \( K_i \) is a prime implicant of all (minimal) conflicts
Ranking Diagnoses by Probability

\[ p(c) = \prod_{y_i = v_{ij} \in c} p(y_i = v_{ij}) \]

Assume Failure and Observation Independence

\[ p(c \mid x_i = v_{ij}, Obs) = \frac{p(x_i = v_{ij} \mid c) p(c \mid Obs)}{p(x_i = v_{ij})} \]

Bayes’ Rule

\[ P(x_i = v_{ij} \mid c) \] estimated using Model:

- If previous Obs, c and \( \Psi \) entails \( z = v \)
  Then \( p(z = v \mid c) = 1 \)

- If previous obs, c and \( \Psi \) entails \( x \neq v \)
  Then \( p(z = v \mid c) = 0 \)

- If \( \Psi \) consistent with all values for \( z \)
  Then \( p(x = v \mid c) \) is based on priors
    - E.g., uniform prior = \( 1/m \) for \( m \) possible values of \( x \)
Diagnosis as Conflict-directed Best First Search

Increasing Cost

Infeasible

Conflict 1

Conflict 2

Feasible

Conflict 3

Enumerating Probable Candidates

Leading Candidate Based on Priors

Generate Candidate

Test Candidate

Consistent?

Yes

Keep

Compute Posterior p

Below Threshold?

Yes

No

Done

Extract Conflict

Sherlock: [de Kleer & Williams, 89]
Mode Estimation is an Optimization Problem

Given:
- System variables $X$ with domain $D_X$
- Mode variables $Y$ with domain $D_Y$
- System model $\Psi(X,Y) : D_X \times D_Y \rightarrow \{\text{True, False}\}$
- Observations $\text{Obs}(X,Y) : D_X \times D_Y \rightarrow \{\text{True, False}\}$

Compute:
- Leading Arg Max $\arg \max_{Y \in D_Y} P(Y \mid \text{Obs})$
- s.t. $\exists X \in D_X . \Psi(X,Y) \wedge \text{OBS}(X,Y)$ is consistent
Generalize to Optimal CSP

Constraint Satisfaction Problem
CSP = <X, D_X,C>
- variables X with domain D_X
- Constraint C(X): D_X → {True,False}

Find X in D_X s.t. C(X) is True

Optimal CSP
OCSP= <Y, g, CSP>
- Decision variables Y with domain D_Y
- Utility function g(Y): D_Y → R
- CSP is over variables <X,Y>

Find Leading arg max g(Y)
Y ∈ D_Y
s.t. ∃ X ∈ D_Y s.t. C(X,Y) is True

Outline
- Review of Diagnosis and Mode Estimation
- Generalizing Mode Estimation to Optimal CSPs
  - Model-based Programming w/o State
  - Mode Estimation as Trajectory Tracking
Example: The model-based program sets \textit{engine = thrusting}, and the deductive controller . . .

- **Mode Estimation**
  - Oxidizer tank
  - Fuel tank
  - Deduces that thrust is off, and the engine is healthy

- **Mode Reconfiguration**
  - Plans actions to open six valves
  - Deduces that a valve failed - stuck closed

- **Mode Reconfiguration**
  - Determines that valves on the backup engine will achieve thrust, and plans needed actions.
**Mode Estimation:**
Select a most likely set of component modes that are consistent with the model and observations

\[
\text{arg min } P_t(Y | \text{Obs}) \\
\text{s.t. } \Psi(X, Y) \land O(m') \text{ is consistent}
\]

**Mode Reconfiguration:**
Select a least cost set of commandable component modes that entail the current goal, and are consistent

\[
\text{arg max } R_t(Y) \\
\text{s.t. } \Psi(X, Y) \text{ entails } G(X, Y) \\
\text{s.t. } \Psi(X, Y) \text{ is consistent}
\]

**OpSat:**

\[
\text{arg min } f(x) \\
\text{s.t. } C(x) \text{ is satisfiable} \\
\text{D(x) is unsatisfiable}
\]
A Simple Concurrent State Model

- A device is described by a set of components communicating through shared variables.
- A component has a set of modes and state variables.
  - Modes are mutually exclusive and collectively exhaustive
  - All components include the unknown mode.
- A mode has a:
  - probability
  - cost
  - state constraint

\[
\begin{align*}
\text{vlv=open} & \Rightarrow \text{Outflow} = M_z^+(\text{inflow}); \\
\text{vlv=stuck open} & \Rightarrow \text{Outflow} = M_z^-(\text{inflow}); \\
\text{vlv=stuck closed} & \Rightarrow \text{Outflow} = 0;
\end{align*}
\]

Simple Mode Estimation

Find most likely modes consistent with observations.

Goal: Left engine on

Observe “no thrust”

Enumerate by decreasing prior probability.
Update by Bayes rule.
**Simple Mode Reconfiguration**

Find allowed states that entail goal.

Enumerate by increasing immediate cost.

**Conflicts Focus MR**

Goal: Achieve Thrust

A conflict, C, is an assignment to a subset of the control variables that entails the negation of the goal.
A conflict, $C$, is an assignment to a *subset* of the control variables that entails the negation of the goal.
**Conflicts Focus MR**

**Goal: Achieve Thrust**

A *conflict, C*, is an assignment to a *subset* of the control variables that entails the negation of the goal.

---

**Performance on Cassini**

Number of components: 80  
Number of clauses: 11,101

<table>
<thead>
<tr>
<th>Failure scenario</th>
<th>MI time (Sparc 5 in sec)</th>
<th>MR time (Sparc 5 in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGA preaim</td>
<td>2.2</td>
<td>1.7</td>
</tr>
<tr>
<td>BPLVD</td>
<td>2.7</td>
<td>2.9</td>
</tr>
<tr>
<td>IRU</td>
<td>1.5</td>
<td>1.6</td>
</tr>
<tr>
<td>EGA burn</td>
<td>2.2</td>
<td>3.6</td>
</tr>
<tr>
<td>ACC</td>
<td>2.5</td>
<td>1.9</td>
</tr>
<tr>
<td>ME too hot</td>
<td>2.4</td>
<td>3.8</td>
</tr>
<tr>
<td>Acc low</td>
<td>5.5</td>
<td>6.1</td>
</tr>
</tbody>
</table>
Outline

- Review of Diagnosis and Mode Estimation
- Generalizing Mode Estimation to Optimal CSPs
  - Model-based Programming w/o State
  - Mode Estimation as Trajectory Tracking

Estimating Dynamic Systems

Given sequence of commands and observations:
1. Infer current distribution of (most likely) states
2. Infer (most likely) trajectories
Hidden Markov Model

- \( X, A, Z \): Finite States, Actions & Observations
- \( T(x,a,x') \): State transition function
  \( T: X \times A \rightarrow \Pi(X) \quad p(x'|a,x) \)
- \( O(x,z) \): Observation function
  \( O: X \rightarrow \Pi(Z) \quad p(z|x) \)
- \( \pi(x) \): Initial state distribution
  \( \pi: \Pi(X) \)

Hidden Markov Models

- \( X = \{\text{Coin}_1, \text{Coin}_2\} \)
- \( Z = \{H, T\} \)
- \( A = \{} \)
- \( T, O, \pi: \)

Observed sequence:
\( h, t, h, h, h, t, h, t, h, h, h, h, h, h \)

Hidden sequence:
\( C_1, C_1, C_1, C_1, C_2, C_2, C_2, C_2, C_1, C_1, C_2, C_2, C_2, C_2 \)
**HMM Belief Tracking**

\[ p_t(x) = p(x_t | z, a, \pi) \]

1) Initialization

\[ p_1(x) = \pi_1(x) \]

2) Induction

Correct \hspace{1cm} Predict

\[ p_t(x_i) = \frac{O(x_i, z_t) \sum_{1 \leq j \leq |X|} T(x_j, a_t, x_i) p_{t-1}(x_j)}{p(z_t | a_t, p_{t-1})} \]

Normalize

**Concurrent Probabilistic Constraint Automata**

component modes…

described by finite domain constraints on variables…

deterministic and probabilistic transitions

cost/reward

**Engine Model**

(thrust = zero) AND (power_in = zero)  
(thrust = zero) AND (power_in = nominal)  
(thrust = full) AND (power_in = nominal)

**Camera Model**

(power_in = zero) AND (shutter = closed)  
(power_in = nominal) AND (shutter = open)
Comparison With Path Planning

- Few variables
- $O(10^4)$ states
- Simple Action & Observation Models

- Hundreds of variables
- $O(10^{150})$ states (modes)
- Propositional Action & Observation Models

How can we avoid enumerating the belief state?

Online Mode Estimation

Track distribution of reachable modes consistent with observations.
Track most likely trajectories.

Left engine on
Observe "no thrust"

K-best Filtering:
Enumerate by decreasing probability.
Online Mode Estimation

- Find feasible next states, given current state distribution, commands and next state observations.

\[ \rho_{S_{i+1}} = \bigwedge_j \left( \rho_{S_i} \land \rho_{S_{ui}} \land \Phi_{j} \land \Psi_{j} \right) \land \rho_{S_{o_{i+1}}} \]

where transition \( \tau_j \) is specified by a conjunction of formulas \( \Phi_{j} \Rightarrow \text{next}(\Psi_{j}) \)

HMM Trajectory Tracking

1) Initialization
\[ \delta_i(x_i) = \pi_i p(z_i | x_i) \]
\[ \psi_i(x_i) = 0 \]

2) Induction
\[ \delta_i(x_i) = O(x_i, z_i) \max_{1 \leq j \leq |X|} \left[ T(x_j, a_j, x_i) \delta_{t-1}(x_j) \right] \]
\[ \psi_i(x_i) = \arg \max_{1 \leq j \leq |X|} \left[ T(x_j, a_j, x_i) \delta_{t-1}(x_j) \right] \]

3) Termination
\[ P^* = \max_{1 \leq i \leq |X|} \left[ \delta_i(x_i) \right] \]
\[ x_t^* = \arg \max_{1 \leq i \leq |X|} \left[ \delta_i(x_i) \right] \]

4) Backtracking
\[ x_t^* = \psi_{t+1}(x_{t+1}^*) \]
Probabilities for Mode Estimation

- Initial state for each component is independent
  \[ \pi(c) = \prod_{v_i = v_{ij} \in c} \pi_i(y_i = v_{ij}) \]

- \( O(x_i, z_t) \) estimated using model as before.

- Component transitions are independent, given previous state and action.
  \[ T(x_j, a_j, x_i) = \prod_k T_k(x_j, a_j, x_{ik}) \]

- \( T_k(x_j, a_j, x_{ik}) = P(\text{transition}) \cdot P(\text{guard_satisfied}) \)
- \( P(\text{guard_satisfied}) \) estimated from model

Enumerating Probable Transitions

For Each Component
Identify Enabled Transitions
And Their Probability

Leading
Candidate Transition
Based on Predicted
Next State Probability

Generate
Candidate Transition

Test Target State
Against Observations

Consistent?

Yes
- Keep
- Compute Posterior P Using Bayes Rule

No
- Extract Conflict

Consistent?

Below Threshold?

Yes
- Done

No
- Below Threshold?
Example

- Valve Driver (VDU) commands valves
- Flow measured at each valve
- VDU may hang, valves may stick shut
- \( \tau \) captures non-determinism of \( T(s,a,s')c \)

![Diagram showing Valve Driver (VDU) commands valves, Flow measured at each valve, VDU may hang, valves may stick shut, \( \tau \) captures non-determinism of \( T(s,a,s')c \).]

Tracking the Most Likely States

- Start with known state(s)
- Track the most likely outcome(s) given observation
- Can be made very fast
  ➔ But most likely state may first appear (a priori) arbitrarily unlikely.
N-Stage Trajectory Tracking
[Kurien & Nayak, 2000]

- Avoid committing to a small number of trajectories
- Build a structure that compactly represents all evolutions
- Generate additional trajectories in order as needed

Temporal Conflicts:
VDU OK in past
V1 open, no V1 flow;
VDU OK in past
V2 open, no V2 flow;

Current Conflicts:
V1 open, no V1 flow;
V2 open, no V2 flow

Graphical Model of Probabilistic Concurrent Constraint Automata

\[ P(\text{tau}=\text{hang}) = p \]
Trajectory Tracking
Framed as an Optimal CSP

• Assignments to $\tau$ capture every system trajectory
• Probability of an assignment is defined to be $\prod_{\text{components}} \prod_{\text{time}} P(\tau_{\text{component},t})$
• Trajectories can be generated in prior probability order

Focusing on Consistent Trajectories

Observe: $\text{Flow}_{v1} = \text{zero}$
noGood = \{ $\tau_{vdu}$=nom, $\tau_{v1,1}$=nom, $\tau_{v1,0}$=nom\}

Observe: $\text{Flow}_{v2} = \text{zero}$
noGood = \{ $\tau_{\text{cmdin}}$=nom, $\tau_{v2,1}$=nom, $\tau_{v2,0}$=nom\}
Online Mode Estimation Demonstrations

- In-situ (e.g., Mars) propellant production
- Testbed in '00.

- Flown on Deep Space One
- Deployed on interferometry testbed

- L2 integrated into IVHM architecture
- Experiment on X34 in '02
- Experiment on X37 in '03

- Integrated into 3T architecture
- Control of bioregenerative life support

Images courtesy of NASA.

Model-based Programming

- To survive decades embedded systems orchestrate complex regulatory and immune systems.
- Future systems will be programmed with models, describing themselves and their environments.
- Runtime kernels will be agile, deducing and planning by solving optimization problems with propositional constraints.
- OpSat problems are solved efficiently by combining conflict-recognition and A* search to leap over leading inconsistent solutions.