Automated Instructor Assistant for Ship Damage Control

Vadim Bulitko
University of Alberta
http://www.cs.ualberta.ca/~bulitko
Outline

× Motivation
× Immersive simulator
× AI Instructor
  • AI expert
  • Explainable AI
## The Problem

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>Damaged</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWII</td>
<td>196 ships</td>
<td>15%</td>
<td></td>
</tr>
<tr>
<td>Falklands</td>
<td>15 ships</td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td>P. Gulf</td>
<td>4 ships</td>
<td>75%</td>
<td></td>
</tr>
</tbody>
</table>

- Minimum response time: ~15 mins
- Fix: 1 hours → days
Scheduling Example: Ship Damage Control

Firefighters

Firemain

Fire
Scheduling Example: Ship Damage Control
Scheduling Example: Ship Damage Control

Firemain repair team

Fire fighters

Difficult to patch → shut the cut-off valve

Low pressure!
Damage Control

EOOW
Plotter
Repair 2
CHENG
Fwd BDS
DCCO
DCA
CIC
Aft BDS
Repair 3
Repair 5
CSMC
Repair 1
EOOW
Problems

× Concurrent crises
× Conflicting needs
× Limited resources
× Overwhelming amount of uncertain and possibly contradictory reports
× Stress
× Real-life training is very limited

× “Small” easy-to-make errors --> big problems
Solution

✗ Step 1: create a realistic immersive system
  - A wide range of damage control scenarios

✗ Step 2: create an AI instructor
  - Critic
  - Advisor
Step 1

× Created DC-Train 2.0
  ∙ a first-principle simulator
  ∙ extended CFAST fire model
  ∙ own code for ship crew
    ∙ own audio
    ∙ video clips from IDCTT
  ∙ own code for electrical/mechanical systems
  ∙ own code for flooding
  ∙ own code for firemain simulation

× Tested at SWOS, Newport, R.I. in 1998
Scenario Specification

× Script/GUI-based
Damage Control Simulator-Trainer
Need a human instructor

× To demonstrate a successful solution
× To explain what went wrong
× To suggest a course of actions on the fly
× To score student’s performance

× Lots of students
× Few instructors...
Solution: Minerva-DCA

✗ Designed an AI agent able to:
  - Achieve an expert-level performance
  - Explain its own actions
  - Observe student’s actions and:
    - Advise
    - Critique
    - Score performance
Problem-Solving Module

- A wide-range rule-based plan generator
- Explicit meta-level reasoning
- Petri Net based lookahead predictor
- Machine-learned state evaluator

- Reactively generate a set of (partial) plans
- Predict & evaluate their outcomes
- Select the most promising plan
An Example

GOALS:

Fix the rupture

ACTIONS:

Shut the valve; patch
Patch and plug in place

SCORES:

no pressure \rightarrow fire spreads \rightarrow explosion in a weapons comp.

Environment Model

low pressure \rightarrow fire gets put out

STATE EVALUATOR

10 mins

\infty mins

Plan Generation

Plan Evaluation/Selection
Plan Generator - 1

× Domain knowledge
  • Rule-based format
  • Easily explainable

\[
\text{ccf}(r012,1,1,[\text{alarm, fire, Where, Time}], 800, [\text{fire, Where, FireClass, discovered, Time}], 0.6, []). \\
\text{ccb}(r012,1,1,[\text{alarm, fire, Where, TimeAlarm}], 800, [\text{fire, Where, FireClass, Status, Time}], 0.6).
\]

**Meaning:** finding alarm indicates hypothesis fire with a confidence of 0.6.
Plan Generator - 2

Strategy knowledge for meta-reasoning

Top-level goals:
- process_hypothesis(Hypothesis)
- process_finding(Finding)
- explore_hypothesis(Hypothesis)
- remove_datum(Datum)

Intermediate-level goals:
- applyrule_backward(Rule)
- applyrule_forward(Rule)
- findout(Datum)
- pursue_hypothesis(Hypothesis)
- test_hypothesis(Hypothesis)

Bottom-level goals:
- perform(Action)
- lookup(Finding)
- remove(Datum)
- conclude(Hypothesis)

mr(pf1, process_finding(F), applyrule_forward(Rule, Hyp)) :-
    finding(F, _),
    red-flag(F),
    ccf(Rule, N, M, F, CF, Hyp, CFC, UL),
    satisfied(F, CF).

Meaning: strategy goal process_finding(F) could be reduced to strategy goal applyrule_forward(Rule, Hyp) if there is a domain rule Rule with F as its condition clause and the clause is satisfied. So if we are processing finding F and there is a rule conditioned on F then try to apply that rule.
Figure 4. A Strategy Chain Example: Order Repair-3 station to investigate compartment 01-300-2 since there is a suspected fire in the compartment; a fire report would confirm/disconfirm the fire hypothesis; and a fire report could be obtained through investigation.
Explainability

× Domain knowledge encodes naval DC doctrines
  • Fire alarm can indicate fire
× Strategy knowledge encodes common sense meta-reasoning
  • If X is important and is suspected and doing Y tests X then do Y
× Both are in a Horn-clause/Prolog-like structure and allow for NL generation
Predictor

Spatial referencing
Temporal referencing

Ignited \iff hot, no_fbs.

\text{ignited}(Y,T') \iff \text{hot}(X,T), \not \text{fbs}(X,R,T),
\text{delay}(T,T',3,4), \text{neighbor}(X,Y).
**Explainability**

- Petri Nets are used for concurrent simulation

- Can crudely predict outcomes of hundreds of scenarios in a fraction of the time it takes to simulate one

- Because of the rule-based predicate-calculus nature, nature can be traced and explained
Evaluator

- In board games (e.g., chess) typically a weighted combination of hand-coded features
- Here, an ANN:
  - Status of all compartments and systems as the input
  - Predicted time to a kill point as the output
  - Learnt from previous scenarios
  - Assumes an average level of DC efforts
Explainability

× Limited to saying something like:

× “Based on the (massive, simulated) historical data, the ship is estimated to last N minutes from here”
Experimental Evaluation

- Minerva-DCA is a decision-making system for automated ship damage control
- Minerva-DCA with a TIPN scheduler was tested at a naval officer training school
- It exhibited a 318% improvement over Navy officers in handling ship crises
Experimental Evaluation

× 160 simulated damage control scenarios in DC-Train 1.0

× Scenario outcomes:
  - **Ship lost** -- a major disaster within first 25 minutes
  - **Ship possibly saved** -- no active fires at the 25-minute mark
  - **Ship saved** -- no active crises at the 25-minute mark

× DCAs:
  - DC students (SWOS, Newport, Rhode Island)
  - Minerva with a TIPN-based scheduler
Experimental Evaluation

× DC students:

- Ship Lost: 39
- Ship Possibly Saved: 28
- Ship Saved: 93
Experimental Evaluation

Minerva with a TIPN-based scheduler:

- Ship Saved: 117
- Ship Possibly Saved: 22
- Ship Lost: 21

+ 318 %
- 46 %
Advising Mode

- Student is solving a scenario
- Minerva-DCA runs as a backseat driver
- Top N actions are presented as an advice
- Clicking on each delivers an explanation
Advising GUI

**Toolbar:** allows to navigate the graphs and control different options

**NL explanation window:** Provides a NL output for an suggested action

**DO list:** recommended actions

Because there is a fire suspected in compartment 3-338-0, the DCA should request permission from the Captain to flood the compartment.

To CO: Request permission to flood 3-338-0
To REPAIR_3: Investigate fire in 3-338-0
To REPAIR_3: Investigate fire in 3-370-0
To CO: Report manned and ready and zebra set throughout the ship
Ready
Critiquing Mode

✗ Student is solving a scenario
✗ Minerva-DCA is a backseat driver
✗ Errors of omission:
  • Student fails to take one of the high-ranked actions
✗ Errors of commission:
  • Student takes a low ranked action
  • Student takes an action that has not been generated but if evaluated has a low score
Critiquing GUI

NL explanation window:
Provides a NL output for a selected critique

Suboptimal actions performed by the supervisor

Actions the supervisor should have taken while overriding the system
Extensions

.Xna Support of alternative courses of action
  A1 = “R2 to investigate HT alarm in X”
  A2 = “R5 to investigate HT alarm in X”
  Student selects A1
  Minerva-DCA selects A2

× If A1 does address the strategy goal behind A2:
  process_hypothesis(fire,X)
× Then A1 is not an error of commission
× And A2 is not an error of omission
Thanks

http://www.cs.ualberta.ca/~bulitko