New Refinement Strategies for Cartesian Abstractions

David Speck\textsuperscript{1,2} \hfill Jendrik Seipp\textsuperscript{2}

\textsuperscript{1}University of Freiburg, Germany
\textsuperscript{2}Linköping University, Sweden
Motivation

1. David Speck, Jendrik Seipp – New Refinement Strategies for Cartesian Abstractions
Motivation
Motivation
Motivation
Motivation
Motivation
New refinement strategies

Counterexample-Guided Cartesian Abstraction Refinement (CEGAR)

- Repeatedly find counterexamples
  - I.e., abstract plans that fail for the concrete task
  - Repair the flaw by splitting a state
New refinement strategies

Counterexample-Guided Cartesian Abstraction Refinement (CEGAR)

- Repeatedly find counterexamples
  - I.e., abstract plans that fail for the concrete task
  - Repair the flaw by splitting a state
- Previously: choose an arbitrary optimal abstract plan
New refinement strategies

Counterexample-Guided Cartesian Abstraction Refinement (CEGAR)

- Repeatedly find counterexamples
  - i.e., abstract plans that fail for the concrete task
  - Repair the flaw by splitting a state
- Previously: choose an arbitrary optimal abstract plan

Contribution

- Consider all optimal abstract plans
- New refinement strategies = flaw + split selection strategies
How can we find all flaws?

Flaw Search

- Consider all optimal abstract plans
- Depth-first search in the concrete transition system
- Consider only $f$-optimal transitions of the abstraction
- Collect all encountered flaws
- Goal state expanded $\leadsto$ optimal concrete plan
- Open list empty $\leadsto$ all flaws found
Flaw selection strategy

**FIRST strategy [SH18]**
- Considers an *arbitrary* optimal abstract plan $\pi$
- Selects the *first* flaw found for $\pi$
- Returns $\pi$ if it works for the concrete task
Flaw selection strategy

**FIRST strategy [SH18]**
- Considers an arbitrary optimal abstract plan $\pi$
- Selects the first flaw found for $\pi$
- Returns $\pi$ if it works for the concrete task

**MINH strategy**
- Considers all optimal abstract plans
- Selects a flaw with the lowest $h$-value $\rightsquigarrow$ close to the goal
- Returns a concrete solution if one exists
Flaw selection strategy

**FIRST strategy [SH18]**
- Considers an arbitrary optimal abstract plan $\pi$
- Selects the first flaw found for $\pi$
- Returns $\pi$ if it works for the concrete task

**MinH strategy**
- Considers all optimal abstract plans
- Selects a flaw with the lowest $h$-value $\leadsto$ close to the goal
- Returns a concrete solution if one exists

**MaxH strategy**
- Considers all optimal abstract plans
- Selects a flaw with the highest $h$-value $\leadsto$ far to the goal
- Returns a concrete solution if no flaw exists
Flaw selection strategy – Batch refinement

- Searching for all flaws in every step can be expensive
  ⇔ Repair several flaws at once
Flaw selection strategy – Batch refinement

- Searching for all flaws in every step can be expensive
- Repair several flaws at once

**Batch strategy**

- Search for all flaws
- Return a concrete plan if the flaw search found one
- Iteratively repairs the flaw with the lowest $h$-value
Flaw selection strategy – Batch refinement

- Searching for all flaws in every step can be expensive
  ⇝ Repair several flaws at once

**BATCH strategy**

- Search for all flaws
- Return a concrete plan if the flaw search found one
- Iteratively repairs the flaw with the lowest $h$-value
- Attention: repairing a flaw can change the abstraction!
- Check if $h$-values of flaws have changed
- Repair all flaws that maintain the $h$-value
Flaw selection strategy – Theoretical results

**FIRST/MaxH** can lead to arbitrarily larger abstractions until a concrete solution is found, compared to **MinH/Batch**.
Flaw selection strategy – Theoretical results

**FIRST/MaxH** can lead to *arbitrarily larger* abstractions until a concrete solution is found, compared to **MinH/BATCH**.
Flaw selection strategy – Theoretical results

**FIRST/MaxH** can lead to *arbitrarily larger* abstractions until a concrete solution is found, compared to **MinH/Batch**.
Flaw selection strategy – Theoretical results

**First/MaxH** can lead to **arbitrarily larger** abstractions until a concrete solution is found, compared to **MinH/Batch**.
Flaw selection strategy – Theoretical results

**FIRST/MaxH** can lead to **arbitrarily larger** abstractions until a concrete solution is found, compared to **MinH/Batch**.
Flaw selection strategy – Theoretical results

**FIRST/MaxH** can lead to *arbitrarily larger* abstractions until a concrete solution is found, compared to **MinH/BATCH**.
Split selection strategy

- Usually many different ways to repair a flaw
- I.e., how to split the abstract state
Split selection strategy

- Usually many different ways to repair a flaw
- I.e., how to split the abstract state

**MaxRefined** strategy [SH18]

- Splits the domain of the variable that has been refined the most
Split selection strategy

- Usually many different ways to repair a flaw
- I.e., how to split the abstract state

**MaxRefined strategy [SH18]**
- Splits the domain of the variable that has been refined the most

**Cover strategy**
- Consider multiple flaws at once
- Chooses split that addresses the most flaws at once
Experiments

- Implemented refinement strategies in Scorpion [Sei18]
- Planning tasks from optimal track of IPCs
- 15 min and 3.5GB for CEGAR
- 30 min and 4GB memory for $A^* + h^{CEGAR}$
# Experiments – Coverage

<table>
<thead>
<tr>
<th>Strategy</th>
<th>MaxRefined</th>
<th>Cover</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST</td>
<td>第一</td>
<td>MINH</td>
</tr>
<tr>
<td>MAXH</td>
<td>345</td>
<td>MINH</td>
</tr>
<tr>
<td>MINH</td>
<td>382</td>
<td>BATCH</td>
</tr>
<tr>
<td>CEGAR</td>
<td>499</td>
<td>534</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MINH</th>
<th>BATCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>393</td>
<td>534</td>
</tr>
</tbody>
</table>
Experiments – Runtime

<table>
<thead>
<tr>
<th>First+MaxRefined</th>
<th>MaxH+MaxRefined</th>
<th>MinH+MaxRefined</th>
<th>MinH+Cover</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Batch+Cover

see legend

Batch+Cover

n.s.
Experiments – Abstraction Size

**Legend:**
- **First+MaxRefined**
- **MaxH+MaxRefined**
- **MinH+MaxRefined**
- **MinH+Cover**

**Graph:**
- The x-axis represents the **Batch+Cover** metric.
- The y-axis represents the n.s. (not significant) metric.
- The graph shows a scatter plot with various markers representing the different strategies.
Experiments – Coverage

<table>
<thead>
<tr>
<th>Strategy</th>
<th>MaxRefined</th>
<th>Cover</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>First</td>
<td>MaxH</td>
</tr>
<tr>
<td>CEGAR</td>
<td>499</td>
<td>345</td>
</tr>
<tr>
<td>$A^* + h^{CEGAR}$</td>
<td>802</td>
<td>780</td>
</tr>
</tbody>
</table>
Experiments – Expansions

- First + MaxRefined
- MaxH + MaxRefined
- MinH + MaxRefined
- MinH + Cover

see legend

Batch + Cover

David Speck, Jendrik Seipp – New Refinement Strategies for Cartesian Abstractions
Experiments – Heuristic Accuracy

- First + MaxRefined
- MaxH + MaxRefined
- MinH + MaxRefined
- MinH + Cover

Batch + Cover

see legend
Conclusion

- New refinement strategies for CEGAR $\rightsquigarrow$ flaw + split selection strategies
- Flaw Search $\rightsquigarrow$ determine all flaws simultaneously
Conclusion

- New refinement strategies for CEGAR \( \rightsquigarrow \) flaw + split selection strategies
- Flaw Search \( \rightsquigarrow \) determine all flaws simultaneously

Findings

- Refine states close to the goal
- Split states such that multiple flaws are repaired at once
- Repair as many flaws as possible in one step (batch)
Conclusion

- New refinement strategies for CEGAR \(\rightarrow\) flaw + split selection strategies
- Flaw Search \(\rightarrow\) determine all flaws simultaneously

Findings
- Refine states close to the goal
- Split states such that multiple flaws are repaired at once
- Repair as many flaws as possible in one step (batch)

Future work
- Compare refinement strategies for multiple Cartesian abstractions