Utilizing Quad-Trees for Efficient Design Space Exploration with Partial Assignment Evaluation

Kai Neubauer\textsuperscript{1}, Philipp Wanko\textsuperscript{2}, Torsten Schaub\textsuperscript{2}, and Christian Haubelt\textsuperscript{1}

\textsuperscript{1}University of Rostock, Germany
\textsuperscript{2}University of Potsdam, Germany
Introduction

- System-level design is becoming more complex
  - Mapping, allocation, and scheduling
  - Heterogeneous processing platforms
- Aim is to find a system implementation that is...
  - valid regarding various constraints
  - optimal regarding quality properties
- Multiple approaches, e.g.:
  - Meta-Heuristics (MOEA, MOPSO, Ant colony, etc.)
  - Formal methods (SAT, ILP, ASP, etc.)
  - Hybrid techniques
Research Issues

- Meta-Heuristics not executed systematically
  - Combining previously found solutions
  - Tend to run into saturation
- Formal symbolic techniques
  - Find all solutions
  - Huge search space
- Problem: Enormous amount of comparisons
  - Fitness vectors stored in an archive
  - Dominance checks for novel solutions
  - Partial Assignment Evaluation
Multi-objective Optimization

- Design space exploration in order to find optimal solutions
- Optimality dependent on quality parameters
  - Often conflicting objectives – Multi-objective optimization Problem (MOOP)
  - Different design alternatives not totally ordered
  - Pareto optimality
Multi-objective Optimization

- Dominance relations between $X = [x_i]$ and $Y = [y_i]$
  - $X$ dominates $Y$
    - $\forall i: x_i \succeq y_i \land \exists i: x_i > y_i$
  - $X$ is dominated by $Y$
    - $\forall i: x_i \preceq y_i \land \exists i: x_i < y_i$
  - $X$ is incomparable to $Y$
    - $\exists i: x_i > y_i \land \exists i: x_i < y_i$
- $X$ is Pareto optimal iff it is not dominated by any other solution
- Minimization is assumed in the following
Acquiring Pareto-Front

Problem Instance

- Limit reached?
  - yes
  - no
- Make decision
  - Complete?
    - yes
    - no
- Prune search space
- Update archive
- Check constraints
  - Complete? yes
  - Complete? no
  - Limit reached? yes
- Done

Complete? yes

Limit reached? yes

Failed
List-based Management

- Non-dominated solutions are saved to archive
- List-based approaches
  - $O(n)$ for 2 dimensions – sorted for both dimensions
List-based Management

- Non-dominated solutions are saved to archive
- List-based approaches
  - $O(n)$ for 2 dimensions – sorted for both dimensions
  - $O(n^m)$ for more dimensions – only first dim. sorted
Tree-based Management

- Tree-based approaches
  - Each node represents one solution
  - Each solution is root to further solutions
  - “Ordered” by some degree
  - Comparing all solutions is unnecessary

![Tree diagram with nodes labeled 12, 16, 19, 32, 36, 18, 14, 26, 21, 13, 37, 20, 20, 21, 15, 18, 35]
Quad-Trees (QT) use b-tree structure
- Each node represents one non-dominated solution
- With m objectives: $2^m$ children
- Children “0” and “m-1” are not saved
  - 0 dominates root
  - m-1 is dominated by root
- Each node is represented by its fitness vector
Quad-Trees

- New solutions may be added in one of the $2^m$ sub-trees
  - *k-successor* (m bit) determines the position a children is inserted
  - States which objective is better (0) or worse (1)
  - Position in m-dimensional coordinate system

\[ X = [2, 2] \]

- Diagram showing the quad-tree structure with binary coordinates.
Updating Quad-Trees

k=001\textsubscript{b}

k=000\textsubscript{b}

k=001\textsubscript{b}

15-Feb-18 Jeju Island, Korea, ASP-DAC 2018
## Comparison: List vs QT

<table>
<thead>
<tr>
<th>List</th>
<th>Quad-Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple implementation</td>
<td>Constant complexity</td>
</tr>
<tr>
<td>Good 2-D performance</td>
<td>Fast dominance test</td>
</tr>
<tr>
<td>Removing is easy</td>
<td>Geometrically ordered</td>
</tr>
<tr>
<td>Bad 3-D+ complexity</td>
<td>Complex implementation</td>
</tr>
<tr>
<td></td>
<td>Removing is hard</td>
</tr>
</tbody>
</table>
QTs For Partial Assignments

Problem Instance

Limit reached?

Make decision

Check constraints and dominance

Complete?

Done

Prune search space

Update archive

Limit reached? yes

Complete? no

Failed

Limit reached? failed

Complete? yes
QTs For Partial Assignments

- Checking partial solutions is expensive
  - Each decision (set of decisions) has to be checked
- No need to check if partial solution dominates any other
  - Check if it is already dominated
  - Archive is only updated for complete solutions
- Expensive operations are only executed once
- Ratio Checking to Updating increases
Evaluation

- Setup:
  - Implementation in Python 2.7
  - Spherical Pareto-Front (2 to 5 Dimensional) – Fig. (a)
  - Dominated solutions – Fig. (b)
  - 50 to 200 partial solutions – Fig. (c)
Evaluation (cont.)

Comparisons

Pareto Front Size vs. # Comparisons

- List
- Quad-Tree
Evaluation (cont.)

Execution Time

Time in s

Pareto Front Size

List  Quad-Tree  Timeout
Conclusion

- Quad-Trees for formal methods with Partial Assignment Evaluation
- Dominance Checks performed for each partial solution
- Quad-Trees offer a fast dominance identification
- Significantly lower number of comparisons

Future Work:
- Balancing algorithms
- Use structural information for steering solving process
References
