Multi-Criteria Optimization for Mapping Programs to Multi-Processors

S. Cotton\textsuperscript{1} O. Maler\textsuperscript{1} J. Legriel\textsuperscript{1,2} S. Saidi\textsuperscript{1,2}

\textsuperscript{1}Verimag

\textsuperscript{2}STMicroelectronics

June 15
Outline

1. Introduction

2. Multi-Criteria optimization using SAT queries
   - Strategy1: Under-approximation with Refinement
   - Strategy2: Distance Reduction

3. Implementation and Experiments

4. Conclusion
Outline

1. Introduction
2. Multi-Criteria optimization using SAT queries
   - Strategy 1: Under-approximation with Refinement
   - Strategy 2: Distance Reduction
3. Implementation and Experiments
4. Conclusion
Motivation: Mapping applications to multicore platforms

- How to exploit efficiently emerging embedded multi-core fabrics?
Motivation: Mapping applications to multicore platforms

- A key factor: Mapping while Optimizing several criteria.
- Some criteria are conflicting: no one optimal solutions, but a set of Pareto-optimal solutions.
Mapping applications to multicore platforms

Example:

- Trade-offs between load balancing and communication cost.
Mapping applications to multicore platforms

Example:

- choice 1: optimizes communication cost but not load balancing.
Example:

- choice 2: optimizes load balancing but not communication cost.
Mapping applications to multicore platforms

Application Model: *task-data graphs*

It is a tuple \( G = (P, d, v) \) where,
- \( P \) is a finite set of tasks,
- \( d : P \rightarrow \mathbb{N} \): workload of each task,
- \( v : P \times P \rightarrow \mathbb{N} \cup \{\bot\} \): communication cost between tasks.
Mapping applications to multicore platforms

Architecture Model:
It is a tuple $E = (M, N, b, \rho)$ where,

- $M$ is a finite number of processors,
- $N \subseteq M \times M$ is a set of communication channels.
- $\rho : M \times M \rightarrow N^*$ is a routing function: associates to each pair of processors $(m, m')$ an acyclic path $(m, m_1), (m_1, m_2), \ldots (m_k, m')$ in the network.

we assume that the routing is fixed.
Mapping applications to multicore platforms

Mapping: assigns tasks to processors
- It is a function $\mu : P \rightarrow M$ with $\mu(p) = m$

Cost functions:
- Workload distribution balance: $W^*$ is the average workload.
  $$\Delta = \sum_{m \in M} |W(m) - W^*|$$
- Communication cost:
  $$C = \sum_{(p, p') : \mu(p) \neq \mu(p')} v(p, p') \times |\rho(\mu(p), \mu(p'))|$$
Multi-Criteria Optimization

- Finding optimal trade-offs.
Multi-Criteria Optimization

- Finding optimal trade-offs.

Domination Relation:

\[ s \prec s' (s \text{ dominates } s') \equiv \forall i \ s_i \leq s'_i \land \exists j \ s_j < s'_j \]
Multi-Criteria Optimization

Pareto-optimal Solutions:

- a point $s$ is *minimal* if it is not dominated by any other point.
- set of all *non dominated incomparable* points are Pareto-optimal.
Multi-Criteria Optimization

Pareto-optimal Solutions:

- a point $s$ is *minimal* if it is not dominated by any other point.
- set of all *non dominated incomparable* points are Pareto-optimal.
Satisfiability is a decision problem,
- A logic formula $\phi$ over a set of boolean variables $X$ is satisfiable if there exists a valuation $\alpha : X \rightarrow \{0, 1\}$ s.t $\phi$ is true: SAT
- If no such assignment exists, then the formula is UNSAT.

SMT (Satisfiability Modulo Theory) extends the problem to NON boolean variables.

Great leap in performance has been achieved for search based methods for satisfiability.
Multi-Criteria Optimization and SAT Solvers

- **Satisfiability** is a decision problem,
  - A logic formula $\phi$ over a set of boolean variables $X$ is **satisfiable** if there exists a valuation $\alpha : X \rightarrow \{0, 1\}$ s.t $\phi$ is true: **SAT**
  - If no such assignment exists, then the formula is **UNSAT**.
- **SMT** (Satisfiability Modulo Theory) extends the problem to NON boolean variables.
- Great leap in performance has been achieved for search based methods for satisfiability.
- **Goal**: test the applicability and performance of such tools to solve optimization problems in the multicore domain.
Outline

1. Introduction

2. Multi-Criteria optimization using SAT queries
   - Strategy 1: Under-approximation with Refinement
   - Strategy 2: Distance Reduction

3. Implementation and Experiments

4. Conclusion
Constrained Optimization Problems have the form:

$$\min c(x) \text{ s.t. } \phi(x)$$

where,

- $x$ is the a vector of decision variables,
- $\phi$ is a set of constraints,
- $c = (c_1, \ldots, c_d)$ is a d-dimensional cost vector.
Multi-Criteria optimization using SAT queries

- Constrained Optimization Problems have the form:

\[ \min c(x) \text{ s.t. } \phi(x) \]

where,
- \( x \) is the a vector of decision variables,
- \( \phi \) is a set of constraints,
- \( c = (c_1, \ldots, c_d) \) is a d-dimensional cost vector.

We optimize the cost by submitting a sequence of SAT queries to the solver of the form:

\[ \exists x \phi(x) \land c(x) < s? \]
One dimensional case

- Sequence of existential queries with decreasing thresholds to arrive to an optimal solution:
One dimensional case

Sequence of existential queries with decreasing thresholds to arrive to an optimal solution:

\[ \exists x \phi(x) \land c(x) < c_1? \]
One dimensional case

- Sequence of existential queries with decreasing thresholds to arrive to an optimal solution:

\[ \exists x \, \phi(x) \land c(x) < c_1? \]

- SAT

0 \quad c_2 \quad c_1 \quad C
One dimensional case

Sequence of existential queries with decreasing thresholds to arrive to an optimal solution:

\[ \exists x \phi(x) \land c(x) < c_2? \]

\[ 0 \quad c_2 \quad c_1 \quad C \]
One dimensional case

- Sequence of existential queries with decreasing thresholds to arrive to an optimal solution:
  \[ \exists x \phi(x) \land c(x) < c_2? \]

  ![Diagram showing sequence of existential queries with decreasing thresholds](Diagram)

  **c_2** is The optimal solution
Multi-dimensional case

- More choices for defining the sequence of queries to find pareto-optimal solutions.
- We experiment 2 strategies for exploring the search space:
  1. Under-approximation with Refinement.
  2. Distance Reduction.
Outline

1. Introduction

2. Multi-Criteria optimization using SAT queries
   - Strategy1: Under-approximation with Refinement
   - Strategy2: Distance Reduction

3. Implementation and Experiments

4. Conclusion
Algorithm Sketch

Algorithm

\[ \delta = [R] \]  
\[ \text{repeat} \]
\[ \text{repeat} \]
\[ \text{query}(\bar{s}) \]  
\[ \text{if } \text{sat} \]
\[ \bar{s} = s \]  
\[ \text{until } \text{unsat} \]  
\[ \delta = \delta \cup \{R_1, R_2\} \]  
\[ \text{query}(\bigvee_i R_i) \]  
\[ \text{if } \text{sat} \]
\[ \bar{s} = s \]  
\[ \text{else } \text{terminate} \]  
\[ \text{until } \text{terminate} \]  

% initially incomparable area is the whole space

% is there a solution \( s \) with cost \( s \leq \bar{s} \)?

% update threshold

% \( \bar{s} \) is a pareto optimal point

% update incomparable area \( \delta \)

% is there a feasible solution \( s \) in \( \bigvee_i (R_i) \)?

% update threshold for next serie of queries

% \( \bar{s} \) is a pareto optimal point

% update threshold for next serie of queries

% \( \bar{s} \) is a pareto optimal point
Algorithm Sketch

Algorithm

\[
\delta = [R] \quad \% \text{initially incomparable area is the whole space}
\]

\textbf{repeat}

\textbf{repeat}

\textbf{query}(\bar{s}) \quad \% \text{is there a solution } s \text{ with cost } s \leq \bar{s}?

\textbf{if sat}

\bar{s} = s \quad \% \text{update threshold}

\textbf{until unsat} \quad \% \bar{s} \text{ is a pareto optimal point}

\delta = \delta \cup \{R_1, R_2\} \quad \% \text{update incomparable area } \delta

\textbf{query}(\bigvee_i R_i) \quad \% \text{is there a feasible solution } s \text{ in } \bigvee_i(R_i)\

\textbf{if sat}

\bar{s} = s \quad \% \text{update threshold for next serie of queries}

\textbf{else terminate}

\textbf{until terminate}
Algorithm

\[ \exists x \; \phi(x) \land c(x) \leq \bar{s} \]
Algorithm

\[ \exists x \ (\phi(x) \land c(x) \leq \bar{s}?) \Rightarrow SAT : s_1 \]
Algorithm

\[ \exists x \; \phi(x) \land c(x) \leq s_1 ? \]
Algorithm

\[ \exists x \ (\phi(x) \land c(x) \leq s_1) \Rightarrow SAT : s_2 \]
Algorithm

\[ \exists x \; \phi(x) \land c(x) \leq s_2? \]
Algorithm

\[ \exists x \ (\phi(x) \land c(x) \leq s_2) \Rightarrow SAT : s_3 \]
Algorithm

\[ \exists x \ (\phi(x) \land c(x) \leq s_3) \]

\[ \Rightarrow \text{UNSAT} \]
Algorithm

\[ \exists x \phi(x) \land c(x) \leq s_3 \Rightarrow UNSAT \]
Algorithm

- Update incomparable search area \( \delta \):
  \[ \delta = \delta \cup \{ R_1, R_2 \} \]
2 strategies to direct the search in incomparable search area $\delta = \bigcup_i R_i$:

- **Union** strategy: query($\bigvee_i R_i$) search directed by the solver.
- **maximum rectangle** strategy: query($\max(R_i)$) direct the search in the largest rectangle.
Algorithm Sketch

Algorithm

\[
\delta = \left[ \right] \quad \% \text{initially incomparable area is empty}
\]

repeat
  repeat
    query(\(\bar{s}\)) \quad \% \text{is there a solution } s \text{ with cost } s \leq \bar{s} ?
    if sat
      \(\bar{s} = s\) \quad \% \text{update threshold}
  until unsat \quad \% \(\bar{s}\) is a pareto optimal point

\(\delta = \delta \cup \{R_1, R_2\}\) \quad \% \text{update incomparable area } \delta

query(\(\bigvee_i R_i\)) \quad \% \text{is there a feasible solution } s \text{ in } \bigvee_i (R_i)?

if sat
  \(\bar{s} = s\) \quad \% \text{update threshold for next serie of queries}
else terminate
until terminate

Cotton, Maler, Legriel, Saidi ()
Multi-Criteria Optimization for Mapping Prog
SIES 2011 23 / 38
Algorithm Sketch

Algorithm

\[
\delta = [ ] \quad \% \text{ initially incomparable area is empty}
\]

repeat

repeat

query(\(\bar{s}\)) \quad \% \text{ is there a solution } s \text{ with cost } s \leq \bar{s}?

if sat

\(\bar{s} = s\) \quad \% \text{ update threshold}

until unsat \quad \% \(\bar{s}\) is a pareto optimal point

\[
\delta = \delta \cup \{R_1, R_2\} \quad \% \text{ update incomparable area } \delta
\]

query(max(\(R_i\))) \quad \% \text{ is there a feasible solution } s \text{ in } max(\(R_i\))? 

if sat

\(\bar{s} = s\) \quad \% \text{ update threshold for next serie of queries}

else terminate

until terminate
Algorithm

\[
\begin{align*}
C_1 & \quad C_2 \\
\bar{s} & \quad s
\end{align*}
\]

Points:
- \(s_1\)
- \(s_2\)
- \(s_3\)
- \(s_4\)
Algorithm
Algorithm

\begin{center}
\begin{tikzpicture}
\draw[step=1cm,very thin,black!30] (-1,-1) grid (4,4);
\fill[red!30] (0,0) rectangle (1,1);
\fill[blue!30] (1,1) rectangle (4,4);
\fill[red!30] (0,1) rectangle (1,2);
\fill[blue!30] (1,2) rectangle (4,4);
\fill[blue!30] (0,2) rectangle (1,4);
\fill[red!30] (0,3) rectangle (1,4);
\node at (0.5,0.5) {$s_5$};
\node at (1.5,1.5) {$s_4$};
\node at (2.5,2.5) {$s_3$};
\node at (3.5,3.5) {$s_1$};
\node at (4.5,4.5) {$\bar{s}$};
\node at (0.5,3.5) {$C_2$};
\node at (1.5,0.5) {$C_1$};
\end{tikzpicture}
\end{center}
Algorithm

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]

\[ \text{Algorithm} \]
Outline

1. Introduction

2. Multi-Criteria optimization using SAT queries
   - Strategy1: Under-approximation with Refinement
   - Strategy2: Distance Reduction

3. Implementation and Experiments

4. Conclusion
Based on [LGCM10]

- Maintain the **distance** between unsat and sat points.
Based on [LGCM10]

- Maintain the **distance** between unsat and sat points.

![Diagram showing the distance between unsat and sat points](image)
Based on [LGCM10]

- Maintain the **distance** between unsat and sat points.
Based on [LGCM10]

- Maintain the **distance** between unsat and sat points.

![Diagram showing distance reduction strategy (C_1 and C_2)]
Based on [LGCM10]

- Maintain the **distance** between unsat and sat points.
Based on [LGCM10]

- Maintain the distance between unsat and sat points.
Multi-Criteria optimization using SAT queries

Strategy 2: Distance Reduction

Distance

- Use the distance $d$ to **direct** the search. 3 variants of how to chose $s$ in $d$
  
  1. Binary search: query with $s = u^* + \frac{1}{2}(d, d)$
  2. Bias towards feasible: query with $s = u^* + \frac{3}{4}(d, d)$
  3. Randomly: ask $u^* + r(d, d)$ where $r$ is randomly chosen in $(0, 1)$
after each query, the distance is recomputed and the new value of distance directs the search in another region of the space.
after each query, the distance is recomputed and the new value of distance directs the search in another region of the space.
after each query, the distance is recomputed and the new value of distance directs the search in another region of the space.
Outline

1. Introduction

2. Multi-Criteria optimization using SAT queries
   - Strategy1: Under-approximation with Refinement
   - Strategy2: Distance Reduction

3. Implementation and Experiments

4. Conclusion
Applications

- synthetic task graphs generated with TGFF tool.
- 2 realistic task graphs derived from image and video filtering applications: TMNR and Demosaicing.
We suppose an architecture with 4 or 8 processors and NoC with a *Spidergon* topology.
SAT Solver

- SMT solver: Z3 developed at Microsoft Research.
- Mapping decision variables are boolean variables, $X_{ij}$: task $i$ is mapped on processor $j$.
- Use of the C API of Z3, and stores the logical context when possible between queries.
Performance and Time Bounded Queries

- for small problems the strategies get to the pareto-optimal points in a reasonable time: (on spidergon4, for both industrial applications, converged to the optimum within 180s)
- for large instances, some satisfiability results become harder to prove, especially around the pareto curve,
Performance and Time Bounded Queries

- for small problems the strategies get to the pareto-optimal points in a reasonable time: (on spidergon4, for both industrial applications, converged to the optimum within 180s)

- for large instances, some satisfiability results become harder to prove, especially around the pareto curve,

- we parametrize the queries with timeouts, and a timed out query is treated as an unsat query.

<table>
<thead>
<tr>
<th>global timeout (sec.)</th>
<th>per-query timeout (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1, 10</td>
</tr>
<tr>
<td>30</td>
<td>5, 10, 30</td>
</tr>
<tr>
<td>60</td>
<td>5, 10, 60</td>
</tr>
<tr>
<td>180</td>
<td>5, 10, 180</td>
</tr>
</tbody>
</table>
Comparing the strategies

We use the **relative volume/area metric**:

- It represents how much each strategy contributes to the global dominated area.

![Diagram showing Strategy 1 with regions s1, s2, and s3]
Comparing the strategies

We use the *relative volume/area metric*:

- It represents how much each strategy contributes to the global dominated area.

\[ \text{Strategy 2} \]

\[ r_2 \]

\[ C_2 \]

\[ r_1 \]

\[ C_1 \]
Comparing the strategies

We use the relative volume/area metric:

- It represents how much each strategy contributes to the global dominated area.

\[ r_1\]

\[ r_2\]

\[ s_1\]

\[ s_2\]

\[ s_3\]

\[ C_1\]

\[ C_2\]

Merge points both strategies
Comparing the strategies

We use the **relative volume/area metric**:

- It represents how much each strategy contributes to the global dominated area.
Comparing the strategies

We use the \textit{relative volume/area} metric:

- It represents how much each strategy contributes to the global dominated area.

![Diagram of global dominated area with strategies s1, s2, s3, and C1, C2]
Comparing the strategies

We use the **relative volume/area metric**:

- It represents how much each strategy contributes to the global dominated area.

![Diagram](image)
Comparing the strategies

We use the relative volume/area metric:

- It represents how much each strategy contributes to the global dominated area.

![Diagram showing the comparison of strategies]

\[ \text{Dominated area Strategy2} \]
Comparing the strategies

General results according to the volume/area metric:

<table>
<thead>
<tr>
<th>tasks</th>
<th>time</th>
<th>t/query</th>
<th>maxrect</th>
<th>union</th>
<th>bin</th>
<th>sat</th>
<th>rand</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>30</td>
<td>5</td>
<td>0.81</td>
<td>0.75</td>
<td>0.85</td>
<td>0.84</td>
<td>0.85</td>
</tr>
<tr>
<td>10</td>
<td>180</td>
<td>5</td>
<td>0.95</td>
<td>0.90</td>
<td>0.99</td>
<td>1.00</td>
<td>0.85</td>
</tr>
<tr>
<td>15</td>
<td>60</td>
<td>10</td>
<td>0.74</td>
<td>0.67</td>
<td>0.79</td>
<td>0.82</td>
<td>0.78</td>
</tr>
<tr>
<td>15</td>
<td>60</td>
<td>60</td>
<td>0.53</td>
<td>0.54</td>
<td>0.40</td>
<td>0.53</td>
<td>0.37</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>5</td>
<td>0.40</td>
<td>0.72</td>
<td>0.77</td>
<td>0.75</td>
<td>0.70</td>
</tr>
<tr>
<td>30</td>
<td>180</td>
<td>5</td>
<td>0.68</td>
<td>0.78</td>
<td>0.93</td>
<td>0.88</td>
<td>0.92</td>
</tr>
<tr>
<td>30</td>
<td>180</td>
<td>10</td>
<td>0.40</td>
<td>0.80</td>
<td>0.91</td>
<td>0.92</td>
<td>0.88</td>
</tr>
<tr>
<td>35</td>
<td>30</td>
<td>5</td>
<td>0.60</td>
<td>0.66</td>
<td>0.72</td>
<td>0.77</td>
<td>0.64</td>
</tr>
<tr>
<td>35</td>
<td>60</td>
<td>5</td>
<td>0.57</td>
<td>0.76</td>
<td>0.86</td>
<td>0.76</td>
<td>0.76</td>
</tr>
<tr>
<td>35</td>
<td>60</td>
<td>10</td>
<td>0.59</td>
<td>0.62</td>
<td>0.69</td>
<td>0.76</td>
<td>0.66</td>
</tr>
<tr>
<td>35</td>
<td>180</td>
<td>5</td>
<td>0.79</td>
<td>0.69</td>
<td>0.90</td>
<td>0.86</td>
<td>0.93</td>
</tr>
<tr>
<td>35</td>
<td>180</td>
<td>10</td>
<td>0.64</td>
<td>0.75</td>
<td>0.91</td>
<td>0.85</td>
<td>0.81</td>
</tr>
<tr>
<td>45</td>
<td>30</td>
<td>5</td>
<td>0.52</td>
<td>0.69</td>
<td>0.66</td>
<td>0.71</td>
<td>0.70</td>
</tr>
<tr>
<td>45</td>
<td>30</td>
<td>10</td>
<td>0.45</td>
<td>0.62</td>
<td>0.60</td>
<td>0.73</td>
<td>0.66</td>
</tr>
<tr>
<td>45</td>
<td>60</td>
<td>10</td>
<td>0.46</td>
<td>0.72</td>
<td>0.71</td>
<td>0.71</td>
<td>0.72</td>
</tr>
<tr>
<td>45</td>
<td>180</td>
<td>10</td>
<td>0.50</td>
<td>0.83</td>
<td>0.82</td>
<td>0.88</td>
<td>0.82</td>
</tr>
</tbody>
</table>
Comparing the strategies

Evolution of the area percentage over execution time:

- around 180s: is a good tradeoff between precision and computation time of these problems.
- max rectangle seems to perform well, the locality of queries take benefit from saving the Z3 logical context.
Conclusion

Outline

1. Introduction

2. Multi-Criteria optimization using SAT queries
   - Strategy1: Under-approximation with Refinement
   - Strategy2: Distance Reduction

3. Implementation and Experiments

4. Conclusion
Conclusion

- We propose several strategies for guiding the solver in choosing queries in the cost space.
- We had to parametrize our search algorithms with a time budget per query.
- It turns out that the choice of this parameter is more relevant than the strategy itself. Unfortunately, it's difficult.

Possible improvements:
- Adapt the timeout per query to the area of search.
- Apply the same strategies to the scheduling problem, which is more complex.