Parallel Backtracking with Answer Memoing for Independent And-Parallelism

Pablo Chico de Guzman\textsuperscript{1} \hspace{1cm} Amadeo Casas\textsuperscript{2} \hspace{1cm} Manuel Carro\textsuperscript{1,3} \hspace{1cm} Manuel V. Hermenegildo\textsuperscript{1,3}

\textsuperscript{1}School of Computer Science, Technical University of Madrid, Spain \hspace{1cm} \textsuperscript{2}Samsung Research, USA \hspace{1cm} \textsuperscript{3}IMDEA Software Institute, Spain

ICLP 2011
Lexington, KY – July 7, 2011
Introduction

Two main sources of parallelism in LP:
- OR-Parallelism: Aurora, MUSE, etc.
- AND-Parallelism: &-Prolog, DDAS, etc.
- Both: (&)ACE, AKL, Andorra-I, EAM, etc.

Classical approach in Independent AND-Parallelism (IAP):
- Conery approach:
  - Copying goals overhead.
  - Nonterminating programs.
- Recomputation + sequential backtracking.
  Saves memory and preserves sequential semantics, but...
  - Recomputation can be inefficient.
  - Sequential backtracking limits parallelism.
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

```
main1(X, Y) :-
g1(X), g2(Y).

main2(X, Y) :-
g1(X) & g2(Y).

g1(X) :- X=1, work(2).
g1(X) :- X=2, work(10).
g2(Y) :- Y=1, work(2).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).
```
Introduction and Motivation

Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

main1(X, Y) :-
g1(X), g2(Y).

main2(X, Y) :-
g1(X) & g2(Y).

g1(X) :- X=1, work(2).
g1(X) :- X=2, work(10).

g2(Y) :- Y=1, work(2).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).

Time_{main1} = \frac{g1}{2}
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

main1(X, Y) :-
g1(X), g2(Y).

main2(X, Y) :-
g1(X) & g2(Y).

g1(X) :- X=1, work(2).
g2(Y) :- Y=1, work(2).
g1(X) :- X=2, work(10).
g2(Y) :- Y=2, work(5).

g2(Y) :- Y=3, work(5).

Time_{main1} = \frac{g1}{2} + \frac{g2}{(2)}
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

\[
\begin{align*}
\text{main1}(X, Y) & : - \\
g1(X), & \text{g2}(Y).
\end{align*}
\]

\[
\begin{align*}
g1(X) & : - \ X=1, \ \text{work}(2). \\
g1(X) & : - \ X=2, \ \text{work}(10).
\end{align*}
\]

\[
\begin{align*}
\text{main2}(X, Y) & : - \\
g1(X) & \ & \text{&} \ & \text{g2}(Y).
\end{align*}
\]

\[
\begin{align*}
g2(Y) & : - \ Y=1, \ \text{work}(2). \\
g2(Y) & : - \ Y=2, \ \text{work}(5). \\
g2(Y) & : - \ Y=3, \ \text{work}(5).
\end{align*}
\]

\[
\begin{align*}
\text{Time}_{\text{main1}} & = \sqrt{2} + \sqrt{2 + 5}
\end{align*}
\]
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

\[
\text{main1}(X, Y) :- \\
g1(X), \quad \text{g2}(Y).
\]

\[
\text{main2}(X, Y) :- \\
g1(X) \quad \& \quad \text{g2}(Y).
\]

\[
g1(X) :- X=1, \quad \text{work}(2). \quad \text{g2}(Y) :- Y=1, \quad \text{work}(2).
\]

\[
g1(X) :- X=2, \quad \text{work}(10). \quad \text{g2}(Y) :- Y=2, \quad \text{work}(5).
\]

\[
g2(Y) :- Y=3, \quad \text{work}(5).
\]

\[
\text{Time}_{\text{main1}} = \underbrace{g1}_{2} + \underbrace{(2 + 5 + 5)}_{g2}
\]
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

\begin{align*}
\text{main1}(X, Y) & :- \\
& \quad g1(X), g2(Y). \\
\text{g1}(X) & :- \ X=1, \ \text{work}(2). \\
\text{g1}(X) & :- \ X=2, \ \text{work}(10). \\
\text{main2}(X, Y) & :- \\
& \quad g1(X) \ \& \ g2(Y). \\
\text{g2}(Y) & :- \ Y=1, \ \text{work}(2). \\
\text{g2}(Y) & :- \ Y=2, \ \text{work}(5). \\
\text{g2}(Y) & :- \ Y=3, \ \text{work}(5). \\
\end{align*}

\[ \text{Time}_{\text{main1}} = \frac{g1}{2} + \left( \frac{g2}{2 + 5 + 5} + \frac{g1}{10} \right) \]
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

\[
\begin{align*}
\text{main1}(X, Y) & : - \\
g1(X), & g2(Y).
\end{align*}
\]

\[
\begin{align*}
g1(X) & : - X=1, \text{work(2)}. \\
g1(X) & : - X=2, \text{work(10)}. \\
g2(Y) & : - Y=1, \text{work(2)}. \\
g2(Y) & : - Y=2, \text{work(5)}. \\
g2(Y) & : - Y=3, \text{work(5)}. \\
\end{align*}
\]

\[
\text{Time}_{\text{main1}} = \frac{g1}{2} + \frac{g2}{2 + 5 + 5} + \frac{g1}{10} + \frac{g2}{2}
\]
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

\[
\begin{align*}
\text{main1}(X, Y) :&= \\
& \quad g1(X), \ g2(Y). \\
& \quad g1(X) :\ X=1, \ \text{work}(2). \\
& \quad g1(X) :\ X=2, \ \text{work}(10).
\end{align*}
\]

\[
\begin{align*}
\text{main2}(X, Y) :&= \\
& \quad g1(X) \ \& \ g2(Y). \\
& \quad g2(Y) :\ Y=1, \ \text{work}(2). \\
& \quad g2(Y) :\ Y=2, \ \text{work}(5). \\
& \quad g2(Y) :\ Y=3, \ \text{work}(5).
\end{align*}
\]

\[
\text{Time}_{\text{main1}} = \frac{g1}{2} + \left(\frac{g2}{2 + 5 + 5}\right) + \frac{g1}{10} + \left(\frac{g2}{2 + 5}\right)
\]
Introduction and Motivation

Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

main1(X, Y) :-
    g1(X), g2(Y).

main2(X, Y) :-
    g1(X) & g2(Y).

g1(X) :- X=1, work(2).
g1(X) :- X=2, work(10).

g2(Y) :- Y=1, work(2).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).

\[
\text{Time}_{\text{main1}} = \frac{g1}{2} + \left(\frac{g2}{2+5+5}\right) + \frac{g1}{10} + \left(\frac{g2}{2+5+5}\right) = 36 \text{ secs.}
\]
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

main1(X, Y) :-
    g1(X), g2(Y).

main2(X, Y) :-
    g1(X) & g2(Y).

g1(X) :- X=1, work(2).
g2(Y) :- Y=1, work(2).
g1(X) :- X=2, work(10).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).

Time\textsubscript{main1} = \frac{g1}{2} + \frac{g2}{(2 + 5 + 5)} + \frac{g1}{10} + \frac{g2}{(2 + 5 + 5)} = 36 \text{ secs.}

Time\textsubscript{main2} = \max(2, 2)
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

```
main1(X, Y) :-
g1(X), g2(Y).

main2(X, Y) :-
g1(X) & g2(Y).

\[
\begin{align*}
g1(X) & :- X=1, \text{work}(2). \\
g1(X) & :- X=2, \text{work}(10).
\end{align*}
\]

\[
\begin{align*}
g2(Y) & :- Y=1, \text{work}(2). \\
g2(Y) & :- Y=2, \text{work}(5). \\
g2(Y) & :- Y=3, \text{work}(5).
\end{align*}
\]

Time_{main1} = \frac{g1}{2} + (2 + 5 + 5) + \frac{g2}{10} + (2 + 5 + 5) = 36 \text{ secs.}

Time_{main2} = \frac{g1\&\ g2}{max(2, 2)} + (5)
Introduction and Motivation

Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

main1(X, Y) :-
g1(X), g2(Y).

main2(X, Y) :-
g1(X) & g2(Y).

g1(X) :- X=1, work(2).
g1(X) :- X=2, work(10).

g2(Y) :- Y=1, work(2).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).

\[
\text{Time}_{\text{main1}} = 2 + (2 + 5 + 5) + 10 + (2 + 5 + 5) = 36 \text{ secs.}
\]

\[
\text{Time}_{\text{main2}} = \max(2, 2) + (5 + 5)
\]
Introduction and Motivation

Classical approach in IAP

**Example (Traditional IAP execution - main1 vs. main2)**

- **main1(X, Y):**
  - g1(X), g2(Y).
  - g1(X) :- X=1, work(2).
  - g1(X) :- X=2, work(10).

- **main2(X, Y):**
  - g1(X) & g2(Y).
  - g2(Y) :- Y=1, work(2).
  - g2(Y) :- Y=2, work(5).
  - g2(Y) :- Y=3, work(5).

- **Time**
  - \( \text{Time}_{\text{main1}} = g_1 \left( \frac{2}{2} + (2 + 5 + 5) \right) + g_2 \left( \frac{10}{10} + (2 + 5 + 5) \right) = 36 \text{ secs.} \)
  - \( \text{Time}_{\text{main2}} = g_1 \left( \frac{2}{\text{max}(2, 2)} + (5 + 5) \right) + g_2 \left( \frac{5}{\text{max}(10, 2)} \right) \)
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

\[
\text{main1}(X, Y) :- \\
g1(X), g2(Y).
\]

\[
\text{main2}(X, Y) :- \\
g1(X) \& g2(Y).
\]

\[
g1(X) :- X=1, work(2).
\]

\[
g1(X) :- X=2, work(10).
\]

\[
g2(Y) :- Y=1, work(2).
\]

\[
g2(Y) :- Y=2, work(5).
\]

\[
g2(Y) :- Y=3, work(5).
\]

\[
\text{Time}_{\text{main1}} = \underbrace{g1}_{2} + \underbrace{g2}_{2 + 5 + 5} + \underbrace{g1}_{10} + \underbrace{g2}_{2 + 5 + 5} = 36 \text{ secs.}
\]

\[
\text{Time}_{\text{main2}} = \underbrace{\text{max}(2, 2)}_{g1 \& g2} + \underbrace{g2}_{5 + 5} + \underbrace{\text{max}(10, 2)}_{g1 \& g2} + \underbrace{g2}_{5}
\]
Classical approach in IAP

Example (Traditional IAP execution - main1 vs. main2)

main1(X, Y) :-
g1(X), g2(Y).

main2(X, Y) :-
g1(X) & g2(Y).

g1(X) :- X=1, work(2).
g2(Y) :- Y=1, work(2).
g1(X) :- X=2, work(10).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).

\[
\text{Time}_{\text{main1}} = \underbrace{g_1}_{2} + \underbrace{(g_2)}_{2+5+5} + \underbrace{g_1}_1 + \underbrace{(g_2)}_{2+5+5} = 36 \text{ secs.}
\]

\[
\text{Time}_{\text{main2}} = \underbrace{\text{max}(g_1, g_2)}_{\text{max}(2, 2)} + \underbrace{g_2}_{5+5} + \underbrace{\text{max}(g_1, g_2)}_{\text{max}(10, 2)} + \underbrace{g_2}_{5+5} = 32 \text{ secs.}
\]

\[
\text{Speedup} = 1.12. \text{ Can we do better?}
\]
Our approach for IAP

- We propose an improved solution to backtracking in IAP, which combines:
  - Parallel backtracking.
  - Answer memoing.
  - Incremental computation of answers.

- We maintain high-level approach [ICLP’08].
  - Easier to maintain and extend.
Our approach for IAP

Example (IAP with parallel backtracking - main1 vs. main2)

```
main1(X, Y) :-
g1(X), g2(Y).

main2(X, Y) :-
g1(X) & g2(Y).
```

```
g1(X) :- X=1, work(2).
g1(X) :- X=2, work(10).
```

```
g2(Y) :- Y=1, work(2).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).
```

- \( \text{Time}_{\text{main1}} = 2 + (2 + 5 + 5) + 10 + (2 + 5 + 5) = 36 \text{ secs.} \)
Our approach for IAP

Example (IAP with parallel backtracking - main1 vs. main2)

main1(X, Y) :-
g1(X), g2(Y).

main2(X, Y) :-
g1(X) & g2(Y).

g1(X) :- X=1, work(2).
g1(X) :- X=2, work(10).

g2(Y) :- Y=1, work(2).
g2(Y) :- Y=3, work(5).

g2(Y) :- Y=2, work(5).

- Time_{main1} = 2 + (2 + 5 + 5) + 10 + (2 + 5 + 5) = 36 secs.
- Time_{main1} with memoization = \( \frac{g1}{2} \)
Our approach for IAP

Example (IAP with parallel backtracking - main1 vs. main2)

main1(X, Y) :-
g1(X), g2(Y).

main2(X, Y) :-
g1(X) & g2(Y).

g1(X) :- X=1, work(2).
g2(Y) :- Y=1, work(2).
g1(X) :- X=2, work(10).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).

- Time\text{main1} = 2 + (2 + 5 + 5) + 10 + (2 + 5 + 5) = 36 \text{ secs.}

- Time\text{main1} \text{ with memoization} = \frac{g1}{2} + \frac{g2}{(2 + 5 + 5)}
Our approach for IAP

Example (IAP with parallel backtracking - main1 vs. main2)

main1(X, Y) :-
g1(X), g2(Y).

g1(X) :- X=1, work(2).
g1(X) :- X=2, work(10).

main2(X, Y) :-
g1(X) & g2(Y).

g2(Y) :- Y=1, work(2).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).

- Time_{main1} = 2 + (2 + 5 + 5) + 10 + (2 + 5 + 5) = 36 secs.
- Time_{main1} with memoization = \(2 + (2 + 5 + 5) + 10\) = 24 secs.
Our approach for IAP

Example (IAP with parallel backtracking - main1 vs. main2)

```
main1(X, Y) :-
g1(X), g2(Y).
```

```
main2(X, Y) :-
g1(X) & g2(Y).
```

```
g1(X) :- X=1, work(2).
g1(X) :- X=2, work(10).
g2(Y) :- Y=1, work(2).
g2(Y) :- Y=2, work(5).
g2(Y) :- Y=3, work(5).
```

- $\text{Time}_{\text{main1}} = 2 + (2 + 5 + 5) + 10 + (2 + 5 + 5) = 36$ secs.
- $\text{Time}_{\text{main1}}$ with memoization = $\underbrace{2}_{g1} + \underbrace{\left(2 + 5 + 5\right)}_{g2} + \underbrace{10}_{g1} = 24$ secs.
- $\text{Time}_{\text{main2}}$ with memoization = $\max(2, 2)$
Our approach for IAP

### Example (IAP with parallel backtracking - main1 vs. main2)

<table>
<thead>
<tr>
<th>main1(X, Y) :-</th>
<th>main2(X, Y) :-</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1(X), g2(Y).</td>
<td>g1(X) &amp; g2(Y).</td>
</tr>
<tr>
<td>g1(X) :- X=1, work(2).</td>
<td>g2(Y) :- Y=1, work(2).</td>
</tr>
<tr>
<td>g1(X) :- X=2, work(10).</td>
<td>g2(Y) :- Y=2, work(5).</td>
</tr>
<tr>
<td>g2(Y) :- Y=3, work(5).</td>
<td></td>
</tr>
</tbody>
</table>

- **Time** 
  - \( \text{Time}_{\text{main1}} = 2 + (2 + 5 + 5) + 10 + (2 + 5 + 5) = 36 \text{ secs.} \)
  
  - **Time** \( \text{main1} \) with memoization = \( 2 \) + \( 2 + 5 + 5 \) + \( 10 \) = 24 secs.

  \[
  \begin{align*}
  g_1 & \quad 2 \\
  g_2 & \quad 2 + 5 + 5 \\
  g_1 & \quad 10
  \end{align*}
  \]

- **Time** \( \text{main2} \) with memoization = \( \max(2, 2) + \max(10, 5 + 5) = 12 \text{ secs.} \)

- Speedup w.r.t. sequential with memoization = 2.00.

- Speedup w.r.t. sequential = 3.00 (superlinear).
Our approach for IAP - Forward execution

- **Forward execution**: similar to classical IAP.
  - If a goal fails for first solution, conjunction fails (no slow-down!).
  - If/when all goals find a solution, execution proceeds.

- **Speculation**: if conjunction not finished yet, additional solutions are sought for goals that already computed an answer (if free agents).
  - Need to stash away generated solutions to continue searching for more answers (which are also saved).

- **Suspension**: when all goals find a solution, any speculative goals are suspended, their state saved, and their first answer reinstalled.
  - New **answers saved in memoing area** to avoid future recomputation.
  - Every new solution is **combined with** previously available solutions.
Our approach for IAP - Backward execution

- **Parallel backtracking**: if more solutions needed, backward execution performed in parallel. Suspended goals resume.

- **Backward** execution: performed over goals on top of the stack.
  - More efficient implementation, i.e., *trapped goals very infrequent*.
  - Solution order differs from sequential execution.
Memoization of Answers

- Goals are independent:
  - Makes sense to store answers to avoid recomputation (we know better how to do this now).

- Answer memoing different/simpler from tabling:
  - Termination not an issue. No need for detecting repeated calls, suspending/resuming consumers, etc.
  - All stored answers are discarded as soon as the conjunction continues after its last answer, or after the conjunction fails.
  - Visibility of stored answers restricted to only the parallel conjunction.
  - Easier case to implement/maintain than tabling.
Memoization of Answers - Implementation

- Answer memoization = saving a combination of heap and trail terms.
  - Trail segment corresponding to execution of parallel goal, and terms created during the goal execution pointed to by these variables.

- Reinstalling an answer consists of copying saved terms back to heap, and updating the trail.
- Memoization has a cost, but it can provide by itself substantial speedups since it avoids recomputation.
Answer Combination

- When last goal pending to generate an answer in a conjunction produces a solution, it needs to be combined (from right to left) with the rest of the answers to continue with forward execution.

Example (Combining answers in a parallel conjunction)

```
main(X, Y, Z) :-
g1(X) & g2(Y) & g3(Z).
```

Assume $g_1$ and $g_2$ have computed two answers, and $g_3$ only one.

Then, $g_3$ finds a new answer $\{Z = 2\}$.
- First combination of answers will be $(a, x, 2)$.
- Next combination will be $(a, y, 2)$.
- Final combinations will be $(b, x, 2)$ and $(b, y, 2)$.
A *ghost* choice point is created with an alternative that retrieves the saved answers and installs the bindings.

- Heap top pointer of ghost choice point points to the current heap top after copying terms, to protect those terms from backtracking for future answer combinations.
- All these copied terms are released when the ghost choice point is eliminated.
Scheduler for the Parallel Backtracking IAP Engine

agent :- work, agent.

work :-
    find_parallel_goal(Handler) ->
      (goal_not_executed(Handler) ->
        (save_init_execution(Handler),
         call_parallel_goal(Handler)
        ;
         move_execution_top(Handler),
         fail)
      ;
      suspend,
      work).

parcall_back(LGoals, NGoals) :-
    fork(PF, NGoals, LGoals, [Handler|LH]),
    (goal_not_executed(Handler) ->
      call_local_goal(Handler, Goal)
    ;
     true
    ),
    look_for_available_goal(LH),
    join(PF).

look_for_available_goal([]) :- !, true.
look_for_available_goal([Handler|LH]) :-
    (goal_available(Handler) ->
      call_local_goal(Handler, Goal)
    ;
     true
    ),
    look_for_available_goal(LH).
Deterministic Parallel Goals

- Machinery proposed can be greatly simplified if goals deterministic.
  - Answer memoization and answer combination are not needed.
  - High-level scheduler can be greatly simplified.

- Some optimizations can be performed dynamically.

- Performance improvements of up to a factor of two in the execution of deterministic benchmarks.
**Deterministic Benchmarks - Sun Fire T2000**

- 8 cores x 4 threads, 8 Gb of memory, average of 10 runs.
- Speedups w.r.t. sequential (unparallelized) execution.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Approach</th>
<th>Number of processors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Fibo</td>
<td>&amp;-Prolog</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>seqback</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>parback</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;det&lt;/sub&gt;</td>
<td><strong>0.96</strong></td>
</tr>
<tr>
<td>MMatrix</td>
<td>&amp;-Prolog</td>
<td><strong>1.00</strong></td>
</tr>
<tr>
<td></td>
<td>seqback</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>parback</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;det&lt;/sub&gt;</td>
<td><strong>0.80</strong></td>
</tr>
<tr>
<td>QSort</td>
<td>&amp;-Prolog</td>
<td><strong>1.00</strong></td>
</tr>
<tr>
<td></td>
<td>seqback</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>parback</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;det&lt;/sub&gt;</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>seqback&lt;sub&gt;GC&lt;/sub&gt;</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;GC&lt;/sub&gt;</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;GCdet&lt;/sub&gt;</td>
<td><strong>0.97</strong></td>
</tr>
</tbody>
</table>
## Nondeterministic Benchmarks - Sun Fire T2000

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Approach</th>
<th>Number of processors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>CheckFiles</td>
<td>seqback&lt;sub&gt;first&lt;/sub&gt;</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>seqback&lt;sub&gt;all&lt;/sub&gt;</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;first&lt;/sub&gt;</td>
<td>3917</td>
</tr>
<tr>
<td></td>
<td>pb_rel&lt;sub&gt;first&lt;/sub&gt;</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;all&lt;/sub&gt;</td>
<td>12915</td>
</tr>
<tr>
<td></td>
<td>pb_rel&lt;sub&gt;all&lt;/sub&gt;</td>
<td>1.00</td>
</tr>
<tr>
<td>Illumination</td>
<td>seqback&lt;sub&gt;first&lt;/sub&gt;</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>seqback&lt;sub&gt;all&lt;/sub&gt;</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;first&lt;/sub&gt;</td>
<td>1120</td>
</tr>
<tr>
<td></td>
<td>pb_rel&lt;sub&gt;first&lt;/sub&gt;</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;all&lt;/sub&gt;</td>
<td>8760</td>
</tr>
<tr>
<td></td>
<td>pb_rel&lt;sub&gt;all&lt;/sub&gt;</td>
<td>1.00</td>
</tr>
<tr>
<td>QSortND</td>
<td>seqback&lt;sub&gt;first&lt;/sub&gt;</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>seqback&lt;sub&gt;all&lt;/sub&gt;</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;first&lt;/sub&gt;</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>parback&lt;sub&gt;all&lt;/sub&gt;</td>
<td>4.29</td>
</tr>
<tr>
<td></td>
<td>pb_rel&lt;sub&gt;all&lt;/sub&gt;</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Conclusions

- Developed a parallel (out-of-order) backtracking approach for IAP, which relies on answer memoization to reuse and combine answers.
- Our approach provides significant performance improvements in the execution of nondeterministic parallel calls, due to the answer memoization mechanism and parallel backtracking.
- Optimizations allow avoiding the overhead for deterministic goals.