Applying Multi-Core Model Checking to Hardware-Software Partitioning in Embedded Systems

Alessandro Trindade, Hussama Ismail, and Lucas Cordeiro

Foz do Iguaçu / PR, 5th November of 2015
Motivation

• Embedded systems: parts in **HW** (↑ speed, ↑$$$$) and other parts in **SW** (↓$$, ↓ speed)

• Most critical step in 1st generation of HW/SW Co-design partitioning

• Model checking: describe the system behavior by a precise and not ambiguous (mathematical) model
  – Early detection of errors
  – Explore all states of a system in a automatic way
    (so instead of finding code violations, we can explore states until it solves the partitioning problem)
Motivation

• Embedded systems: parts in **HW** (↑ speed, ↑$$$$) and other parts in **SW** (↓$, ↓ speed)
• Most critical step in 1st generation of HW/SW Co-design partitioning

• Model checking: describe the system behavior by a precise and not ambiguous (mathematical) model
  – Early detection of errors
  – Explore all states of a system in a automatic way
    (so instead of finding code violations, we can explore states until it solves the partitioning problem)
Objectives

- Use OpenMP (Open Multi-Processing API) support to perform multi-core model checking
- Create and improve algorithms to implement the proposed technique
- Perform experimental evaluation over benchmarks
- Compare our approach with ILP (Integer Linear Programming) and GA (Genetic Algorithm) using MATLAB

Apply multi-core model checking based on satisfiability modulo theories (SMT) to solve the HW/SW partitioning
Objectives

Apply multi-core model checking based on satisfiability modulo theories (SMT) to solve the HW/SW partitioning

- Use OpenMP (Open Multi-Processing API) support to perform multi-core model checking
- Create and improve algorithms to implement the proposed technique
- Perform experimental evaluation over benchmarks
- Compare our approach with ILP (Integer Linear Programming) and GA (Genetic Algorithm) using MATLAB
Objectives

- Use OpenMP (Open Multi-Processing API) support to perform multi-core model checking
- Create and improve algorithms to implement the proposed technique
- Perform experimental evaluation over benchmarks
- Compare our approach with ILP (Integer Linear Programming) and GA (Genetic Algorithm) using MATLAB

Apply multi-core model checking based on satisfiability modulo theories (SMT) to solve the HW/SW partitioning
Objectives

Apply multi-core model checking based on satisfiability modulo theories (SMT) to solve the HW/SW partitioning

- Use OpenMP (Open Multi-Processing API) support to perform multi-core model checking
- Create and improve algorithms to implement the proposed technique
- Perform experimental evaluation over benchmarks
- Compare our approach with ILP (Integer Linear Programming) and GA (Genetic Algorithm) using MATLAB
Objectives

Apply multi-core model checking based on satisfiability modulo theories (SMT) to solve the HW/SW partitioning

- Use OpenMP (Open Multi-Processing API) support to perform multi-core model checking
- Create and improve algorithms to implement the proposed technique
- Perform experimental evaluation over benchmarks
- Compare our approach with ILP (Integer Linear Programming) and GA (Genetic Algorithm) using MATLAB
Optimization

• Find the maximum or minimum value of a function
  – Minimize the effort and maximize the benefit

• There is not a unique method to solve all the problems

• Most popular technique: LP (Linear Programming)
  – Integer Linear Programming
  – Binary Linear Programming

• Heuristics Algorithms: GA (Genetic Algorithm) can solve more complex problems faster
  – Drawback: it may not find the global minimum/maximum (i.e., the optimal result)
Optimization

• Find the maximum or minimum value of a function
  – Minimize the effort and maximize the benefit

• There is not a unique method to solve all the problems

• Most popular technique: LP (Linear Programming)
  – Integer Linear Programming
  – Binary Linear Programming

• Heuristics Algorithms: GA (Genetic Algorithm) can solve more complex problems faster
  – Drawback: it may not find the global minimum/maximum (i.e., the optimal result)
Optimization

• Find the maximum or minimum value of a function
  – Minimize the effort and maximize the benefit

• There is not a unique method to solve all the problems

• Most popular technique: LP (Linear Programming)
  – Integer Linear Programming
  – Binary Linear Programming

• Heuristics Algorithms: GA (Genetic Algorithm) can solve more complex problems faster
  – Drawback: it may not find the global minimum/maximum (i.e., the optimal result)
Optimization

• Find the maximum or minimum value of a function
  – Minimize the effort and maximize the benefit

• There is not a unique method to solve all the problems

• Most popular technique: LP (Linear Programming)
  – Integer Linear Programming
  – Binary Linear Programming

• Heuristics Algorithms: GA (Genetic Algorithm) can solve more complex problems faster
  – Drawback: it may not find the global minimum/maximum (i.e., the optimal result)
Mathematical Modeling

Informal Model (Assumptions)

• There is only one software context and only one hardware context
  – Each component must be mapped into one of these two contexts.

• The software component implementation has a software cost associated (running time)

• The hardware component implementation has a hardware cost associated (area, heat dissipation or energy consumption)

• Premisses:
  – The hardware is significantly faster than software;
  – The running time of hardware is zero;
  – If two components are mapped to the same context, there is no overhead of communication between them.
Mathematical Modeling

Informal Model (Assumptions)

• There is only one software context and only one hardware context
  – Each component must be mapped into one of these two contexts.

• The software component implementation has a software cost associated (running time)

• The hardware component implementation has a hardware cost associated (area, heat dissipation or energy consumption)

• Premisses:
  – The hardware is significantly faster than software;
  – The running time of hardware is zero;
  – If two components are mapped to the same context, there is no overhead of communication between them.
Mathematical Modeling

Informal Model (Assumptions)

- There is only one software context and only one hardware context
  - Each component must be mapped into one of these two contexts.
- The software component implementation has a software cost associated (running time)
- The hardware component implementation has a hardware cost associated (area, heat dissipation or energy consumption)
- Premisses:
  - The hardware is significantly faster than software;
  - The running time of hardware is zero;
  - If two components are mapped to the same context, there is no overhead of communication between them.
Mathematical Modeling

Informal Model (Assumptions)

- There is only one software context and only one hardware context
  - Each component must be mapped into one of these two contexts.

- The software component implementation has a software cost associated (running time)

- The hardware component implementation has a hardware cost associated (area, heat dissipation or energy consumption)

- Premisses:
  - The hardware is significantly faster than software
  - The running time of hardware is zero
  - If two components are mapped to the same context, there is no overhead of communication between them
Mathematical Modeling

Formal Model

- Task graph $G = (V, E)$
- Vertices $V = \{x_1, x_2, \ldots, x_n\}$: nodes are the components of the system to be partitioned (context)
- Each node $x_i$: has the hardware cost $h(x_i)$ and the software cost $s(x_i)$
  - $\$\text{HW}$ (area, heat dissipation, energy consumption)
  - $\$\text{SW}$ (execution time)
- Edges $(E)$ represent communication between the components
- $c(x_i, x_j)$: represents the communication cost between $x_i$ and $x_j$ if they are in different contexts
- The HW-SW partitioning $P$ has:
  \[ \text{\textbf{H}}_P = \sum h_i \text{ (hardware cost)} \]
  \[ \text{\textbf{S}}_P = \sum s_i + \sum c(x_i, x_j) \text{ (software cost)} \]
Mathematical Modeling

Formal Model

- Task graph $G = (V, E)$
- Vertices $V = \{x_1, x_2, \ldots, x_n\}$: nodes are the components of the system to be partitioned (context)
- Each node $x_i$ has the hardware cost $h(x_i)$ and the software cost $s(x_i)
  - $HW (area, heat dissipation, energy consumption)
  - $SW (execution time)
- Edges (E) represent communication between the components
- $c(x_i, x_j)$: represents the communication cost between $x_i$ and $x_j$ if they are in different contexts
- The HW-SW partitioning $P$ has:
  - $H_P = \sum h_i$ (hardware cost)
  - $S_P = \sum s_i + \sum c(x_i, x_j)$ (software cost)
Mathematical Modeling

Formal Model

- Task graph $G = (V, E)$
- Vertices $V = \{x_1, x_2, \ldots, x_n\}$: nodes are the components of the system to be partitioned (context)
- Each node $x_i$: has the hardware cost $h(x_i)$ and the software cost $s(x_i)$
  - $HW$ (area, heat dissipation, energy consumption)
  - $SW$ (execution time)
- Edges $(E)$ represent communication between the components
- $c(x_i, x_j)$: represents the communication cost between $x_i$ and $x_j$ if they are in different contexts
- The HW-SW partitioning $P$ has:
  - $H_P = \sum h_i$ (hardware cost)
  - $S_P = \sum s_i + \sum c(x_i, x_j)$ (software cost)

Example: 10 nodes & 13 edges
Mathematical Modeling

Formal Model

- Task graph $G = (V, E)$
- Vertices $V = \{x_1, x_2, \ldots, x_n\}$: nodes are the components of the system to be partitioned (context)
- Each node $x_i$ has the hardware cost $h(x_i)$ and the software cost $s(x_i)$
  - $\$\text{HW}$ (area, heat dissipation, energy consumption)
  - $\$\text{SW}$ (execution time)
- Edges $(E)$ represent communication between the components
- $c(x_i, x_j)$ represents the communication cost between $x_i$ and $x_j$ if they are in different contexts
- The HW-SW partitioning $P$ has:
  - $H_P = \sum h_i$ (hardware cost)
  - $S_P = \sum s_i + \sum c(x_i, x_j)$ (software cost)

Example: 10 nodes & 13 edges
Mathematical Modeling

Formal Model

- Task graph $G = (V, E)$
- Vertices $V = \{x_1, x_2, \ldots, x_n\}$: nodes are the components of the system to be partitioned (context)
- Each node $x_i$ has the hardware cost $h(x_i)$ and the software cost $s(x_i)$
  - $\$ \text{HW}$ (area, heat dissipation, energy consumption)
  - $\$ \text{SW}$ (execution time)
- Edges $(E)$ represent communication between the components
- $c(x_i, x_j)$: represents the communication cost between $x_i$ and $x_j$ if they are in different contexts
- The HW-SW partitioning $P$ has:
  - $H_P = \sum h_i$ (hardware cost)
  - $S_P = \sum s_i + \sum c(x_i, x_j)$ (software cost)
This paper focuses on the case where the initial software cost is given ($S_0$).

We want $S_P < S_0$ and the minimal necessary hardware cost to resolve the problem. (The complexity is NP-Hard)

### Mathematical Modeling

- **Introduction**
- **Objectives**
- **Background**
- **Partitioning**
- **Results**
- **Conclusions**
- **Future Work**

**Model Checker** (ESBMC)

**ILP**

**GA**

MATLAB – Optimization Toolbox
Bounded Model Checking

- Basic Idea: given a transition system M, check negation of a given property \( \varphi \) up to given depth k

- Translated into a VC \( \psi \) such that: \( \psi \) is satisfiable iff \( \varphi \) has counterexample (steps until the violation) of max. depth k

- BMC has been applied successfully to verify (embedded) software since early 2000's. In 2014, Alessandro used BMC perform HW-SW partitioning.
ESBMC (Model Checker)

- ESBMC (Efficient SMT-Based Context-Bounded Model Checker) is a model checker for ANSI-C and C++ source code
  - Check overflows, pointer safety, memory leaks, arrays bounds, atomicity, etc.
- Uses Satisfiability Modulo Theories (SMT) (addition to Boolean Satisfiability)
- SMT Solvers as back-end to decrease software complexity

Architecture:
ESBMC Architecture

#include<stdio.h>
#include<assert.h>

int main()
{
    int i = 0;
    for (i=0; i<50; i++)
    {
        assert(i%8==0);
        assert(i%8==0);
        assert(i%8==0);
    }
    i=0
    assert(i%8==0)
    i=1
    assert(i%8==0)
    i=2
    assert(i%8==0)
    i=3
    assert(i%8==0)
}

C := [i0 := 0 ^ i1 := 1 ^ i2 := 2]
P := [i0%8 == 0 ^ i1%8 == 0 ^ i2%8 == 0]
# ESBMC Architecture

```
#include<stdio.h>
#include<assert.h>

int main(){
  int i = 0;
  for (i=0; i<50; i++){
    assert(i%8==0);
    i++;
  }
}
```

```
C := [i0 := 0 ^ i1 := 1 ^ i2 := 2]
P := [i0%8 == 0 ^ i1%8 == 0 ^ i2%8 == 0]
```

```
k=3 (bound)
```
ESBMC Architecture

```
#include<stdio.h>
#include<assert.h>

int main(){
    int i = 0;
    for (i=0; i<50;i++){
        assert(i%8==0);
    }
}
```

$k=3$ (bound)
ESBMC Architecture

```c
#include<stdio.h>
#include<assert.h>

int main(){
    int i = 0;
    for (i=0; i<50;i++){
        assert(i%8==0);
    }
}
```

The CFG of the program:

\[ k = 3 \text{ (bound)} \]
ESBMC Architecture

```
#include<stdio.h>
#include<assert.h>

int main(){
    int i = 0;
    for (i=0; i<50; i++){
        assert(i%8==0);
    }
}
```

```
C :=
\[
\begin{bmatrix}
i_0 & := & 0 \\
i_1 & := & 1 \\
i_2 & := & 2
\end{bmatrix}
\]

P :=
\[
\begin{bmatrix}
i_0 \% 8 & := & 0 \\
i_1 \% 8 & := & 0 \\
i_2 \% 8 & := & 0
\end{bmatrix}
\]
```
# ESBMC Architecture

```c
#include<stdio.h>
#include<assert.h>

int main(){
    int i = 0;
    for (i=0; i<50;i++){
        assert(i%8==0);
    }
}
```

C := \[
\begin{cases}
i0 := 0 \\
i1 := 1 \\
i2 := 2
\end{cases}
\]

P := \[
\begin{cases}
i0\%8 = 0 \\
i1\%8 = 0 \\
i2\%8 = 0
\end{cases}
\]
```c
#include<stdio.h>
#include<assert.h>

int main(){
    int i = 0;
    for (i=0; i<50; i++){
        assert(i%8==0);
    }
}
```
ESBMC Architecture

```c
#include<stdio.h>
#include<assert.h>

int main()
{
    int i = 0;
    for (i=0; i<50; i++)
    {
        assert(i%8==0);
    }
}
```

Equations size is proportional to the system complexity
(if the equation has many math expressions, then it results in higher verification time)

Introduction
Objectives
Background
Partitioning
Results
Conclusions
Future Work
ESBMC Architecture

```c
#include<stdio.h>
#include<assert.h>

int main(){
    int i = 0;
    for (i=0; i<50; i++){
        assert(i%8==0);
    }
}
```

Currently, SMT solvers doesn’t support parallelism

C := \[
\begin{bmatrix}
  i0 := 0 \\
i1 := 1 \\
i2 := 2
\end{bmatrix}
\]

P := \[
\begin{bmatrix}
i0\%8 == 0 \\
i1\%8 == 0 \\
i2\%8 == 0
\end{bmatrix}
\]
Open Multi-Processing

- **OpenMP (API)** is a set of directives for parallel programming
  - Support for C/C++, and Fortran
  - Support for different operating systems (Windows, Linux, Mac OSX, HP-UX)

- **Use the *fork-join* model**
  - Threads are managed by the API
  - User customizes the execution

- **Compiler directive based:**

  ```
  int k;
  #pragma omp parallel for
  for (k = 0; k < 10; k++)
    a[k] = 2*a[k] ;
  ```
ESBMC for Optimization

• The first algorithm in ANSI-C for ESBMC solves optimization problems

```
01 Initialize variables
02 Declare number of nodes and edges
03 Declare hardware cost of each node as array (h)
04 Declare software cost of each node as array (s)
05 Declare communication cost of each edge (c)
06 Declare the initial software cost (S₀)
07 Declare transposed incidence matrix graph G (E)
08 Define the solutions variables (xᵢ) as Boolean
09 main {
10   For TipH = 0 to Hmax do {
11     Populate xᵢ with nondeterministic/test values
12     Calculate s(1 − x) + c * |Ex| and store at variable
13     Requirement insured by ASSUME (variable ≤ S₀)
14     Calculate Hp cost based on value tested of xᵢ
15    Violation check with ASSERT (Hp > TipH)
16   }
17 }
```

These variables are declared as matrices or vectors

This loop can generate many math equations. It is hard for the SMT solver

We have CPU’s available Why not parallelize?
Multi-Core ESBMC approach

- Solution: use of OpenMP as front-end of ESBMC
- Use fork-join model provided by OpenMP
- OpenMP API creates N different instances:
  - Instead of trying to solve the partitioning problem just once, it creates N different problems with different TipH values of hardware cost
- If a violation occurs then the optimal value was found. The threads are finished
- `/esbmc-parallel <filename.c> <hmin_value> <Hmax>`
Multi-Core ESBMC approach

- **Solution**: use of OpenMP as front-end of ESBMC
- **Use** fork-join model provided by OpenMP
- **OpenMP API** creates N different instances:
  - Instead of trying to solve the partitioning problem just once, it creates N different problems with different TipH values of hardware cost
  - If a violation occurs then the optimal value was found. The threads are finished

```
/esbmc-parallel <filename.c> <hmin_value> <Hmax>
```
Multi-Core ESBMC approach

- Solution: use of OpenMP as front-end of ESBMC
- Use fork-join model provided by OpenMP
- OpenMP API creates N different instances:
  - Instead of trying to solve the partitioning problem just once, it creates N different problems with different TipH values of hardware cost
- If a violation occurs then the optimal value was found. The threads are finished
- /esbmc-parallel <filename.c> <hmin_value> <Hmax>
Multi-Core ESBMC approach

- Solution: use of OpenMP as front-end of ESBMC
- Use fork-join model provided by OpenMP
- OpenMP API creates N different instances:
  - Instead of trying to solve the partitioning problem just once, it creates N different problems with different TipH values of hardware cost
- If a violation occurs then the optimal value was found. The threads are finished

/esbmc-parallel <filename.c> <hmin_value> <Hmax>
Multi-Core ESBMC approach

- Solution: use of OpenMP as front-end of ESBMC
- Use fork-join model provided by OpenMP
- OpenMP API creates N different instances:
  - Instead of trying to solve the partitioning problem just once, it creates N different problems with different TipH values of hardware cost
- If a violation occurs then the optimal value was found. The threads are finished

```
/esbmc-parallel <filename.c> <hmin_value> <Hmax>
```
Multi-Core ESBMC approach

- Solution: use of OpenMP as front-end of ESBMC
- Use fork-join model provided by OpenMP
- OpenMP API creates N different instances:
  - Instead of trying to solve the partitioning problem just once, it creates N different problems with different TipH values of hardware cost
- If a violation occurs then the optimal value was found. The threads are finished
- /esbmc-parallel <filename.c> <hmin_value> <Hmax>
Multi-Core ESBMC approach

- Solution: use of OpenMP as front-end of ESBMC
- Use fork-join model provided by OpenMP
- OpenMP API creates N different instances:
  - Instead of trying to solve the partitioning problem just once, it creates N different problems with different TipH values of hardware cost
- If a violation occurs then the optimal value was found. The threads are finished

```
/esbmc-parallel <filename.c> <hmin_value> <Hmax>
```
Multi-Core ESBMC approach

- Solution: use of OpenMP as front-end of ESBMC
- Use fork-join model provided by OpenMP
- OpenMP API creates N different instances:
  - Instead of trying to solve the partitioning problem just once, it creates N different problems with different TipH values of hardware cost
- If a violation occurs then the optimal value was found. The threads are finished

```
/esbmc-parallel <filename.c> <hmin_value> <Hmax>
```
Multi-Core ESBMC approach

• Solution: use of OpenMP as front-end of ESBMC
• Use fork-join model provided by OpenMP
• OpenMP API creates N different instances:
  – Instead of trying to solve the partitioning problem just once, it creates N different problems with different TipH values of hardware cost
• If a violation occurs then the optimal value was found. The threads are finished

• /esbmc-parallel <filename.c> <hmin_value> <Hmax>
Experimental Evaluation

• Set up
  – Desktop with 64-bit Ubuntu 14.04 LTS, 15GB of RAM and i7 Intel (8-cores) processor with 3.40 GHz of clock
  – ESBMC v1.24
  – SMT solver: Boolector v. 2.0.1
  – MathWorks MATLAB R2013a (GA and ILP)
  – Time out (TO) = 7200 sec
  – Memory out (MO) = 15GB

• Use 7 benchmarks (with different number of nodes)
• Compare with ESBMC, ESBMC Multi-Core, ILP, and GA
• Each time is the average of three measured times
  – (92% of statistical confidence)
Experimental Evaluation

• Set up
  – Desktop with 64-bit Ubuntu 14.04 LTS, 15GB of RAM and i7 Intel (8-cores) processor with 3.40 GHz of clock
  – ESBMC v1.24
  – SMT solver: Boolector v. 2.0.1
  – MathWorks MATLAB R2013a (GA and ILP)
  – Time out (TO) = 7200 sec
  – Memory out (MO) = 15GB

• Use 7 benchmarks (with different number of nodes)
• Compare with ESBMC, ESBMC Multi-Core, ILP, and GA
• Each time is the average of three measured times
  – (92% of statistical confidence)
Experimental Evaluation

• **Set up**
  – Desktop with 64-bit Ubuntu 14.04 LTS, 15GB of RAM and i7 Intel (8-cores) processor with 3.40 GHz of clock
  – ESBMC v1.24
  – SMT solver: Boolector v. 2.0.1
  – MathWorks MATLAB R2013a (GA and ILP)
  – Time out (TO) = 7200 sec
  – Memory out (MO) = 15GB

• **Use 7 benchmarks (with different number of nodes)**
• **Compare with ESBMC, ESBMC Multi-Core, ILP, and GA**
• **Each time is the average of three measured times**
  – (92% of statistical confidence)
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td>Edges</td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td>$S_0$</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
<tr>
<td><strong>Exact Solution</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>13820</td>
<td>876</td>
</tr>
<tr>
<td>Sp</td>
<td>19</td>
<td>4</td>
<td>19</td>
<td>46</td>
<td>533</td>
<td>4231</td>
<td>297</td>
</tr>
<tr>
<td><strong>ILP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>649</td>
<td>1806</td>
<td>TO</td>
<td>5429</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>-</td>
<td>876</td>
</tr>
<tr>
<td><strong>GA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>340</td>
<td>2050</td>
<td>1372</td>
<td>5000</td>
</tr>
<tr>
<td>Error</td>
<td>13%</td>
<td>0%</td>
<td>29%</td>
<td>2%</td>
<td>-7%</td>
<td>-38%</td>
<td>-28%</td>
</tr>
<tr>
<td><strong>ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>30</td>
<td>314</td>
<td>325</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Multicore ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>1609</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>ESBMC Relative Speedup</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>54</td>
<td>47</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th></th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td>Edges</td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td>$S_0$</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Exact Solution</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>13820</td>
<td>876</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sp</td>
<td>19</td>
<td>4</td>
<td>19</td>
<td>46</td>
<td>533</td>
<td>4231</td>
<td>297</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ILP</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time(s)</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>649</td>
<td>1806</td>
<td>TO</td>
<td>5429</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>-</td>
<td>876</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GA</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time(s)</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>340</td>
<td>2050</td>
<td>1372</td>
<td>5000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>13%</td>
<td>0.0%</td>
<td>29.0%</td>
<td>2%</td>
<td>-7%</td>
<td>-38%</td>
<td>-28%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ESBMC</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time(s)</td>
<td>30</td>
<td>314</td>
<td>325</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Multi-core ESBMC</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time(s)</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>1609</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ESBMC Relative Speedup</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>54</td>
<td>47</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th></th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td>Edges</td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td>$S_0$</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Hp</th>
<th>Sp</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact Solution</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>13820</td>
<td>876</td>
</tr>
<tr>
<td>Sp</td>
<td>19</td>
<td>4</td>
<td>19</td>
<td>46</td>
<td>533</td>
<td>4231</td>
<td>297</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Time(s)</th>
<th>Error</th>
<th>Hp</th>
<th>Sp</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ILP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>2</td>
<td>7</td>
<td>17</td>
<td>4</td>
<td>17</td>
<td>4</td>
<td>17</td>
<td>4</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>19</td>
<td>31</td>
<td>47</td>
<td>31</td>
<td>47</td>
<td>31</td>
<td>47</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Time(s)</th>
<th>Error</th>
<th>Hp</th>
<th>Sp</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>Error</td>
<td>13%</td>
<td>0,0%</td>
<td>29,0%</td>
<td>2%</td>
<td>-7%</td>
<td>-38%</td>
<td>-28%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Time(s)</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>30</td>
<td>314</td>
<td>325</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Time(s)</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
<th>Hs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multi-core ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>1609</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

|                | ESBMC Relative Speedup | |
|----------------|------------------------| |
| **ESBMC Relative Speedup** | 14 | 54 | 47 | |
## Results

<table>
<thead>
<tr>
<th></th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nodes</strong></td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td><strong>Edges</strong></td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td><strong>$S_0$</strong></td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
<tr>
<td><strong>Exact Solution</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>13820</td>
<td>876</td>
</tr>
<tr>
<td>Sp</td>
<td>19</td>
<td>4</td>
<td>19</td>
<td>46</td>
<td>533</td>
<td>4231</td>
<td>297</td>
</tr>
<tr>
<td><strong>ILP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>649</td>
<td>1806</td>
<td>TO</td>
<td>5429</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>-</td>
<td>876</td>
</tr>
<tr>
<td><strong>GA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>340</td>
<td>2050</td>
<td>1372</td>
<td>5000</td>
</tr>
<tr>
<td>Error</td>
<td>13%</td>
<td>0.0%</td>
<td>29.0%</td>
<td>2%</td>
<td>-7%</td>
<td>-38%</td>
<td>-28%</td>
</tr>
<tr>
<td><strong>ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>30</td>
<td>314</td>
<td>325</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Multicore ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>1609</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>ESBMC Relative Speedup</strong></td>
<td>14</td>
<td>54</td>
<td>47</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th></th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nodes</strong></td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td><strong>Edges</strong></td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
</tbody>
</table>

### Exact Solution

<table>
<thead>
<tr>
<th></th>
<th>Hp</th>
<th>Sp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time(s)</strong></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>13%</td>
<td>0,0%</td>
</tr>
</tbody>
</table>

### ILP

<table>
<thead>
<tr>
<th></th>
<th>Hp</th>
<th>Sp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time(s)</strong></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>13%</td>
<td>0,0%</td>
</tr>
</tbody>
</table>

### GA

<table>
<thead>
<tr>
<th></th>
<th>Hp</th>
<th>Sp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time(s)</strong></td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>13%</td>
<td>0,0%</td>
</tr>
</tbody>
</table>

### ESBMC

<table>
<thead>
<tr>
<th></th>
<th>Hp</th>
<th>Sp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time(s)</strong></td>
<td>30</td>
<td>314</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>13%</td>
<td>0,0%</td>
</tr>
</tbody>
</table>

### Multi-core ESBMC

<table>
<thead>
<tr>
<th></th>
<th>Hp</th>
<th>Sp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time(s)</strong></td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>13%</td>
<td>0,0%</td>
</tr>
</tbody>
</table>

### ESBMC Relative Speedup

<table>
<thead>
<tr>
<th></th>
<th>Hp</th>
<th>Sp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time(s)</strong></td>
<td>14</td>
<td>54</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>13%</td>
<td>0,0%</td>
</tr>
</tbody>
</table>

**Software Partitioned Cost (solution)**

- CRC32: 25 nodes, 32 edges, $S = 20$
- Patricia: 21 nodes, 48 edges, $S = 10$
- Dijkstra: 26 nodes, 69 edges, $S = 20$
- Clustering: 150 nodes, 331 edges, $S = 50$
- RC6: 329, 448, 600
- Fuzzy: 261, 422, 4578
- Mars: 417, 422, 300

**HP**

- ILP: 15, 47, 31, 241, 692, 13820, 876
- GA: 15, 47, 31, 241, 692, -

**Sp**

- ILP: 19, 4, 19, 46, 533, 4231
- GA: 19, 4, 19, 46, 533, 4231

**Error**

- ILP: 13%, 0,0%, 29,0%
- GA: 13%, 0,0%, 29,0%
- ESBMC: 13%, 0,0%, 29,0%

**ESBMC Relative Speedup**

- 14, 54, 47

**MO**

- ILP: MO, MO, MO, MO, MO
- GA: MO, MO, MO, MO, MO
- ESBMC: MO, MO, MO, MO, MO

**TO**

- ILP: TO, TO, TO, TO, TO
- GA: TO, TO, TO, TO, TO
- ESBMC: TO, TO, TO, TO, TO
### Results

<table>
<thead>
<tr>
<th></th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td>Edges</td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td>$S_0$</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>13820</td>
<td>876</td>
</tr>
<tr>
<td>Sp</td>
<td>19</td>
<td>4</td>
<td>19</td>
<td>46</td>
<td>533</td>
<td>4231</td>
<td>297</td>
</tr>
<tr>
<td><strong>ILP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>649</td>
<td>1806</td>
<td>TO</td>
<td>5429</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>-</td>
<td>876</td>
</tr>
<tr>
<td><strong>GA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>340</td>
<td>2050</td>
<td>1372</td>
<td>5000</td>
</tr>
<tr>
<td>Error</td>
<td>13%</td>
<td>0,0%</td>
<td>29,0%</td>
<td>2%</td>
<td>-7%</td>
<td>-38%</td>
<td>-28%</td>
</tr>
<tr>
<td><strong>ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>30</td>
<td>314</td>
<td>325</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Multicore ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time(s)</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>1609</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>Hp</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ESBMC Relative Speedup</td>
<td>14</td>
<td>54</td>
<td>47</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td>Edges</td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td>$S_0$</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
<tr>
<td>$S_p$</td>
<td>19</td>
<td>4</td>
<td>19</td>
<td>46</td>
<td>533</td>
<td>4231</td>
<td>297</td>
</tr>
</tbody>
</table>

### ILP
- **Time(s)**: 2 1 2 649 1806 TO 5429
- **Hp**: 15 47 31 241 692 - 876

### GA
- **Time(s)**: 7 7 9 340 2050 1372 5000
- **Error**: 13% 0,0% 29,0% 2% -7% -38% -28%

### ESBMC
- **Time(s)**: 30 314 325 MO MO MO MO
- **Hp**: 15 47 31 - - - -

### Multi-core ESBMC
- **Time(s)**: 2 6 7 1609 TO TO TO
- **Hp**: 15 47 31 241 - - -

### ESBMC Relative Speedup
- **Best Performance**: 14 54 47 - - -
## Results

<table>
<thead>
<tr>
<th></th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td>Edges</td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td>$S_0$</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
</tbody>
</table>

**Exact**

<table>
<thead>
<tr>
<th></th>
<th>Hp</th>
<th>Sp</th>
<th>Time(s)</th>
<th>Hp</th>
<th>Sp</th>
<th>Time(s)</th>
<th>Hp</th>
<th>Sp</th>
<th>Time(s)</th>
<th>Hp</th>
<th>Sp</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC32</td>
<td>15</td>
<td>19</td>
<td>2</td>
<td>649</td>
<td>46</td>
<td>340</td>
<td>15</td>
<td>19</td>
<td>2</td>
<td>15</td>
<td>47</td>
<td>340</td>
</tr>
<tr>
<td>Patricia</td>
<td>47</td>
<td>4</td>
<td>1</td>
<td>1806</td>
<td>533</td>
<td>2050</td>
<td>47</td>
<td>4</td>
<td>1</td>
<td>15</td>
<td>47</td>
<td>340</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>31</td>
<td>19</td>
<td>2</td>
<td>241</td>
<td>46</td>
<td>2050</td>
<td>31</td>
<td>19</td>
<td>2</td>
<td>241</td>
<td>46</td>
<td>2050</td>
</tr>
<tr>
<td>Clustering</td>
<td>241</td>
<td>46</td>
<td>2</td>
<td>692</td>
<td>533</td>
<td>2050</td>
<td>241</td>
<td>46</td>
<td>2</td>
<td>692</td>
<td>533</td>
<td>2050</td>
</tr>
<tr>
<td>RC6</td>
<td>13820</td>
<td>4231</td>
<td>2</td>
<td>876</td>
<td>297</td>
<td>5000</td>
<td>13820</td>
<td>4231</td>
<td>2</td>
<td>876</td>
<td>297</td>
<td>5000</td>
</tr>
<tr>
<td>Fuzzy</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mars</td>
<td>876</td>
<td>297</td>
<td>2</td>
<td>5429</td>
<td>5000</td>
<td>5000</td>
<td>876</td>
<td>297</td>
<td>2</td>
<td>5429</td>
<td>5000</td>
<td>5000</td>
</tr>
</tbody>
</table>

**GA**

<table>
<thead>
<tr>
<th></th>
<th>Time(s)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC32</td>
<td>7</td>
<td>13%</td>
</tr>
<tr>
<td>Patricia</td>
<td>7</td>
<td>0,0%</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>9</td>
<td>29,0%</td>
</tr>
<tr>
<td>Clustering</td>
<td>340</td>
<td>2%</td>
</tr>
<tr>
<td>RC6</td>
<td>2050</td>
<td>-7%</td>
</tr>
<tr>
<td>Fuzzy</td>
<td>1372</td>
<td>-38%</td>
</tr>
<tr>
<td>Mars</td>
<td>5000</td>
<td>-28%</td>
</tr>
</tbody>
</table>

**ESBMC**

<table>
<thead>
<tr>
<th></th>
<th>Time(s)</th>
<th>Hp</th>
<th>MO</th>
<th>Hp</th>
<th>MO</th>
<th>Hp</th>
<th>MO</th>
<th>Hp</th>
<th>MO</th>
<th>Hp</th>
<th>MO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC32</td>
<td>30</td>
<td>15</td>
<td>-</td>
<td>47</td>
<td>-</td>
<td>31</td>
<td>-</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Patricia</td>
<td>314</td>
<td>15</td>
<td>-</td>
<td>47</td>
<td>-</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>325</td>
<td>15</td>
<td>-</td>
<td>47</td>
<td>-</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Clustering</td>
<td>MO</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RC6</td>
<td>MO</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fuzzy</td>
<td>MO</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mars</td>
<td>MO</td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Multi-core ESBMC**

<table>
<thead>
<tr>
<th></th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC32</td>
<td>2</td>
</tr>
<tr>
<td>Patricia</td>
<td>6</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>7</td>
</tr>
<tr>
<td>Clustering</td>
<td>1609</td>
</tr>
<tr>
<td>RC6</td>
<td>TO</td>
</tr>
<tr>
<td>Fuzzy</td>
<td>TO</td>
</tr>
<tr>
<td>Mars</td>
<td>TO</td>
</tr>
</tbody>
</table>

**ESBMC Relative Speedup**

<table>
<thead>
<tr>
<th></th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC32</td>
<td>-</td>
</tr>
<tr>
<td>Patricia</td>
<td>-</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>-</td>
</tr>
<tr>
<td>Clustering</td>
<td>-</td>
</tr>
<tr>
<td>RC6</td>
<td>-</td>
</tr>
<tr>
<td>Fuzzy</td>
<td>-</td>
</tr>
<tr>
<td>Mars</td>
<td>-</td>
</tr>
</tbody>
</table>
# Results

The table below summarizes the results for different algorithms and metrics:

<table>
<thead>
<tr>
<th></th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nodes</strong></td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td><strong>Edges</strong></td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td><strong>$S_0$</strong></td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
<tr>
<td><strong>Exact Solution</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hp</strong></td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>13820</td>
<td>876</td>
</tr>
<tr>
<td><strong>Sp</strong></td>
<td>19</td>
<td>4</td>
<td>19</td>
<td>46</td>
<td>533</td>
<td>4231</td>
<td>297</td>
</tr>
<tr>
<td><strong>ILP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Time (s)</strong></td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>649</td>
<td>1806</td>
<td>TO</td>
<td>5429</td>
</tr>
<tr>
<td><strong>Hp</strong></td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>692</td>
<td>-</td>
<td>876</td>
</tr>
<tr>
<td><strong>Time (s)</strong></td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>340</td>
<td>2050</td>
<td>1372</td>
<td>5000</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>13%</td>
<td>0.0%</td>
<td>29.0%</td>
<td>2%</td>
<td>-7%</td>
<td>-38%</td>
<td>-28%</td>
</tr>
<tr>
<td><strong>ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Time (s)</strong></td>
<td>30</td>
<td>314</td>
<td>325</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
<td>MO</td>
</tr>
<tr>
<td><strong>Hp</strong></td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Multi-core ESBMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Time (s)</strong></td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>1609</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td><strong>Hp</strong></td>
<td>15</td>
<td>47</td>
<td>31</td>
<td>241</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>ESBMC Relative Speedup</strong></td>
<td>14</td>
<td>54</td>
<td>47</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The table shows the performance metrics for various algorithms: CRC32, Patricia, Dijkstra, Clustering, RC6, Fuzzy, and Mars. The metrics include the number of nodes and edges, time (s) for exact and ILP solutions, and error. The worst performance is highlighted for each metric.
## Results

<table>
<thead>
<tr>
<th></th>
<th>CRC32</th>
<th>Patricia</th>
<th>Dijkstra</th>
<th>Clustering</th>
<th>RC6</th>
<th>Fuzzy</th>
<th>Mars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>25</td>
<td>21</td>
<td>26</td>
<td>150</td>
<td>329</td>
<td>261</td>
<td>417</td>
</tr>
<tr>
<td>Edges</td>
<td>32</td>
<td>48</td>
<td>69</td>
<td>331</td>
<td>448</td>
<td>422</td>
<td>600</td>
</tr>
<tr>
<td>$S_0$</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>600</td>
<td>4578</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Exact Solution</th>
<th>ILP</th>
<th>GA</th>
<th>Multi-core ESBMC</th>
<th>ESBMC Relative Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$H_p$</td>
<td>$S_p$</td>
<td>Time(s)</td>
<td>$H_p$</td>
<td>Time(s)</td>
</tr>
<tr>
<td>CRC32</td>
<td>15</td>
<td>19</td>
<td>2</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>Patricia</td>
<td>47</td>
<td>19</td>
<td>1</td>
<td>47</td>
<td>19</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>31</td>
<td>4</td>
<td>2</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>Clustering</td>
<td>241</td>
<td>46</td>
<td>2</td>
<td>241</td>
<td>46</td>
</tr>
<tr>
<td>RC6</td>
<td>692</td>
<td>533</td>
<td>649</td>
<td>692</td>
<td>533</td>
</tr>
<tr>
<td>Fuzzy</td>
<td>13820</td>
<td>4231</td>
<td>1806</td>
<td>13820</td>
<td>4231</td>
</tr>
<tr>
<td>Mars</td>
<td>876</td>
<td>297</td>
<td>TO</td>
<td>876</td>
<td>297</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>13%</td>
<td>0,0%</td>
<td>29,0%</td>
<td>2%</td>
<td>-7%</td>
</tr>
<tr>
<td><strong>Time(s)</strong></td>
<td>340</td>
<td>2050</td>
<td>1372</td>
<td>5000</td>
<td>MO</td>
</tr>
<tr>
<td><strong>Speedup over</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ESBMC Relative</strong></td>
<td>15</td>
<td>14</td>
<td>54</td>
<td>47</td>
<td>-</td>
</tr>
</tbody>
</table>

**Note:** The table compares the performance of different algorithms, including CRC32, Patricia, Dijkstra, Clustering, RC6, Fuzzy, and Mars, in terms of nodes, edges, $S_0$, and time (in seconds) for Exact Solution, ILP, and GA. The table also includes an ESBMC Relative Speedup column for comparison.
Conclusions

• 1<sup>st</sup> generation of co-design:
  – Above 400 nodes: none
  – Until 400 nodes: ILP
  – Until 150 nodes: ESBMC
  – GA (error issues)

• ILP e GA: easier to use but ESBMC: no cost (BSD license)
• MC-ESBMC has better performance than Sequential ESBMC
  (speedup from 14 until 54 and no memory out)
• 150 nodes is a realistic problem? All depends on the granularity of
  problem modeling

Future Work

• ESBMC: study the possibilities to decrease the time to solution
  (solver included)
• Use of ESBMC to more complex types of architecture, including more
  then one CPU (2<sup>nd</sup> generation of co-design)
Conclusions

• 1st generation of co-design:
  – Above 400 nodes: none
  – Until 400 nodes: ILP
  – Until 150 nodes: ESBMC
  – GA (error issues)

• ILP e GA: easier to use but ESBMC: no cost (BSD license)
• MC-ESBMC has better performance than Sequential ESBMC (speedup from 14 until 54 and no memory out)
• 150 nodes is a realistic problem? All depends on the granularity of problem modeling

Future Work

• ESBMC: study the possibilities to decrease the time to solution (solver included)
• Use of ESBMC to more complex types of architecture, including more then one CPU (2nd generation of co-design)
Conclusions

• 1st generation of co-design:
  – Above 400 nodes: none
  – Until 400 nodes: ILP
  – Until 150 nodes: ESBMC
  – GA (error issues)

• ILP e GA: easier to use but ESBMC: no cost (BSD license)
• MC-ESBMC has better performance than Sequential ESBMC (speedup from 14 until 54 and no memory out)
• 150 nodes is a realistic problem? All depends on the granularity of problem modeling

Future Work

• ESBMC: study the possibilities to decrease the time to solution (solver included)
• Use of ESBMC to more complex types of architecture, including more then one CPU (2nd generation of co-design)
Conclusions

• 1\textsuperscript{st} generation of co-design:
  – Above 400 nodes: none
  – Until 400 nodes: ILP
  – Until 150 nodes: ESBMC
  – GA (error issues)

• ILP e GA: easier to use but ESBMC: no cost (BSD license)
• MC-ESBMC has better performance than Sequential ESBMC (speedup from 14 until 54 and no memory out)
• 150 nodes is a realistic problem? All depends on the granularity of problem modeling

Future Work

• ESBMC: study the possibilities to decrease the time to solution (solver included)
• Use of ESBMC to more complex types of architecture, including more then one CPU (2\textsuperscript{nd} generation of co-design)
Conclusions

• 1\textsuperscript{st} generation of co-design:
  – Above 400 nodes: none
  – Until 400 nodes: ILP
  – Until 150 nodes: ESBMC
  – GA (error issues)

• ILP e GA: easier to use but ESBMC: no cost (BSD license)
• MC-ESBMC has better performance than Sequential ESBMC (speedup from 14 until 54 and no memory out)
• 150 nodes is a realistic problem? All depends on the granularity of problem modeling

Future Work

• ESBMC: study the possibilities to decrease the time to solution (solver included)
• Use of ESBMC to more complex types of architecture, including more than one CPU (2\textsuperscript{nd} generation of co-design)
Thank you for your attention!

Contacts:
- alessandro.b.trindade@gmail.com
- hussamaismail@gmail.com
- lucascordeiro@ufam.edu.br