Gergely Kovásznai, Csaba Biró and Balázs Erdélyi

IoT Research Institute
Eszterhazy Karoly University
Eger, Hungary
https://iot.uni-eszterhazy.hu/en

SMT 2018
July 12, 2018
Oxford, UK
Introduction

- Work in progress
- Introducing our OMT solver called **PULI**
- **PULI** ⇒ Hungarian shepherd dog you can control by shouting specific verbal commands
- **PULI** solver ⇒ you can control the solving process by exploiting specific information on the problem to solve
Outline

- How all this began? ⇒ WSN optimization
- Previous experiments with OMT solvers on WSN optimization
- Idea behind Puli ⇒ resource function, regression analysis
- Monotonous problems
- Benchmarks and experiments on WSN optimization
- Benchmarks and experiments on the Knapsack problem
- Conclusion
Wireless Sensor Networks (WSNs) use small, inexpensive, self-powered devices that can sense their environment.

- For agriculture, industry, security, traffic monitoring, military, etc.

- WSN = a mesh of sensor nodes + a set of target points to monitor

- The aim ⇒ To maximize the lifetime of a WSN while respecting certain security and dependability constraints.
Wireless Sensor Networks (WSNs) use small, inexpensive, self-powered devices that can sense their environment.

- For agriculture, industry, security, traffic monitoring, military, etc.

**WSN = a mesh of sensor nodes + a set of target points to monitor**

**The aim ⇒ To maximize the lifetime** of a WSN while respecting certain security and dependability constraints.
WSNs are supposed to respect certain security and dependability constraints, such as

**Coverage constraint.** (mandatory)
Each target point must be covered by \( K \geq 1 \) sensor nodes.

**Evasive constraint.** (optional)
A node must not stay awake for more than \( E \geq 1 \) consecutive time intervals.

**Moving target constraint.** (optional)
A critical target point must not be covered by the same node for more than \( M \geq 1 \) consecutive time intervals.
Figure: Sleep/wake-up scheduling of sensor nodes for coverage with $K = 2$ and evasive constraint with $E = 2$. The active nodes (blue dots) are monitoring the target points (green dots).

- Applied OMT solvers to generate sleep/wake-up scheduling that maximizes a WSN’s lifetime.
- Generated SMT-LIB benchmarks.
- Experimented with OptiMathSAT, Z3 and SYMBA.
- We concluded that “OptiMathSAT provides the most stable performance and scales the best”.

Gergely Kovásznai, Csaba Biró and Balázs Erdélyi

Puli – A Problem-Specific OMT solver
Investigated WSNs’ further aspects that make OMT solving more challenging:

- WSN density: 40-50%, 60-70%, 80-90%
- Constraint settings: all on, evasive off, moving target off, evasive moving off

Generated SMT-LIB benchmarks and experimented with OptiMathSAT

Most challenging benchmarks ⇒ density 80-90% and evasive moving off
Puli’s Idea

- Develop an OMT solver that can adapt to the problem to solve.
  - Main objective ⇒ To be able to solve WSN problems faster over QF_UFLIA.

- Define a “resource function” over the problem.
  We introduced two solver-specific options to SMT-LIB:
  - opt-resource-fun ⇒ To define the resource function.
  - opt-resource-target ⇒ To define target value for the resource function.

- Use regression analysis to guess the optimum.

![Optimization Graph](image-url)
Puli’s Idea

- Develop an OMT solver that can adapt to the problem to solve.
  - Main objective ⇒ To be able to solve WSN problems faster over QF_UFLIA.
- Define a “resource function” over the problem.
  We introduced two solver-specific options to SMT-LIB:
  - opt-resource-fun ⇒ To define the resource function.
  - opt-resource-target ⇒ To define target value for the resource function.
- Use regression analysis to guess the optimum.

![Graph showing resource vs. time with an optimum point]
Puli’s Idea

- Develop an OMT solver that can adapt to the problem to solve.
  - Main objective ⇒ To be able to solve WSN problems faster over QF_UFLIA.
- Define a “resource function” over the problem.
  - We introduced two solver-specific options to SMT-LIB:
    - `opt-resource-fun` ⇒ To define the resource function.
    - `opt-resource-target` ⇒ To define target value for the resource function.
- Use regression analysis to guess the optimum.

![Graph showing the relationship between time and resource with an Optimum point marked.]
Puli’s Idea

- Develop an OMT solver that can adapt to the problem to solve.
  - Main objective ⇒ To be able to solve WSN problems faster over QF_UFLIA.

- Define a “resource function” over the problem.
  We introduced two solver-specific options to SMT-LIB:
  - opt-resource-fun ⇒ To define the resource function.
  - opt-resource-target ⇒ To define target value for the resource function.

- Use regression analysis to guess the optimum.
opt-resource-fun ⇒ An estimation of how long the network will be operational from now on.

\[
\sum_{i=1}^{n} L_i - \sum_{i=1}^{n} \sum_{t=1}^{T} w_{i,t}
\]

where

- \( n \geq 1 \): number of sensor nodes.
- \( L_i \): the lifetime of the \( i^{th} \) node.
- \( T \): the WSN’s lifetime.
- \( w_{i,t} \): Boolean variable that denotes if the \( i^{th} \) node is awake at the \( t^{th} \) time interval.

opt-resource-target ⇒ 0
opt-resource-fun $\Rightarrow$ An estimation of how long the network will be operational from now on.

$$\sum_{i=1}^{n} L_i - \sum_{i=1}^{n} \sum_{t=1}^{T} w_{i,t}$$

where

$n \geq 1$: number of sensor nodes.

$L_i$: the lifetime of the $i^{th}$ node.

$T$: the WSN’s lifetime.

$w_{i,t}$: Boolean variable that denotes if the $i^{th}$ node is awake at the $t^{th}$ time interval.

opt-resource-target $\Rightarrow 0$
1: \textbf{procedure} \textsc{Maximization}(lb)  
2: \hspace{1em} T \leftarrow lb  
3: \hspace{1em} \textbf{while} true \textbf{ do}  
4: \hspace{2em} \textbf{if} \ \textsc{LoadResult}(T) = \text{SAT} \textbf{ then}  
5: \hspace{3em} \text{increment} \ T  
6: \hspace{2em} \textbf{else if} \ \textsc{LoadResult}(T) = \text{UNSAT} \textbf{ then}  
7: \hspace{3em} \textbf{if} \ T = lb \textbf{ then}  
8: \hspace{4em} \text{return} \ null  
9: \hspace{3em} \textbf{end if}  
10: \hspace{3em} \text{decrement} \ T  
11: \hspace{2em} \textbf{else}  
12: \hspace{3em} \textit{smtlib} \leftarrow \textsc{GenerateSmtlib}(T)  
13: \hspace{3em} \textbf{(result,} \ T, \ resource) \leftarrow \textsc{SmtSolve}(\textit{smtlib})  
14: \hspace{3em} \textsc{SaveResult}(T, \ result)  
15: \hspace{3em} \textbf{if} \ result = \text{SAT} \textbf{ then}  
16: \hspace{4em} \textbf{if} \ \textsc{LoadResult}(T + 1) = \text{UNSAT} \textbf{ then}  
17: \hspace{5em} \text{return} \ T  
18: \hspace{4em} \textbf{end if}  
19: \hspace{4em} \textsc{SaveResourcePoint}(T, \ resource)  
20: \hspace{3em} \textit{regression} \leftarrow \textsc{RegressionOnResourcePoints}()  
21: \hspace{3em} T' \leftarrow \text{minimum root of} \ \textit{regression} \ \text{where} \ T' > T  
22: \hspace{3em} \textbf{if} \ \textsc{ConditionForJump}(T, T') \textbf{ then}  
23: \hspace{4em} \ T \leftarrow \textsc{Jump}(T, T')  
24: \hspace{4em} \textbf{end if}  
25: \hspace{2em} \textbf{end if}  
26: \hspace{1em} \textbf{end while}  
27: \hspace{1em} \textbf{end procedure}
Monotonous maximization problem $\Rightarrow$ As incrementing the objective function value, all the SMT instances are SAT until exceeding the optimum.

$\text{GenerateSmtlib}(T)$ adds an assertion on the value of the objective function $f_{\text{OBJ}}$ as follows:

- For non-monotonous problems $\Rightarrow f_{\text{OBJ}} \geq T$
- For monotonous problems $\Rightarrow f_{\text{OBJ}} = T$

Results in a significant speed-up.

WSN optimization is monotonous.
Experiments with WSN Benchmarks

- Network grid of $600 \times 600$ meters.
- 10 sensor nodes, 4 target points.
- 20 instances in each of the 12 benchmarks
  - WSN density: 40-50%, 60-70%, 80-90%
  - Constraint settings: all on, evasive off, moving target off, evasive moving off
- 1200 seconds wall clock time limit.
- 3 GB memory limit.
- 3.6 GHz 8-core CPU with 8 GB memory.
### Experiments with WSN Benchmarks – 40-50% density

<table>
<thead>
<tr>
<th>Constraint settings</th>
<th>#SAT/UNSAT</th>
<th>#TO</th>
<th>Optimum</th>
<th>Runtime</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Puli’s linear search</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>19/0</td>
<td>1</td>
<td>30.5</td>
<td>63.5</td>
<td>75.2</td>
</tr>
<tr>
<td>evasive off</td>
<td>20/0</td>
<td>0</td>
<td>35.5</td>
<td>3.5</td>
<td>66.45</td>
</tr>
<tr>
<td>moving target off</td>
<td>20/0</td>
<td>0</td>
<td>48.2</td>
<td>6.4</td>
<td>72.6</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>20/0</td>
<td>0</td>
<td>53.0</td>
<td>7.95</td>
<td>63.65</td>
</tr>
<tr>
<td><strong>Puli’s binary search</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>19/0</td>
<td>1</td>
<td>30.5</td>
<td>63.7</td>
<td>71</td>
</tr>
<tr>
<td>evasive off</td>
<td>20/0</td>
<td>0</td>
<td>35.5</td>
<td>3.5</td>
<td>62.05</td>
</tr>
<tr>
<td>moving target off</td>
<td>20/0</td>
<td>0</td>
<td>48.2</td>
<td>5.8</td>
<td>66.4</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>20/0</td>
<td>0</td>
<td>53.0</td>
<td>3.65</td>
<td>57.05</td>
</tr>
<tr>
<td><strong>OptiMathSAT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>19/0</td>
<td>1</td>
<td>30.5</td>
<td>173.9</td>
<td>430.6</td>
</tr>
<tr>
<td>evasive off</td>
<td>20/0</td>
<td>0</td>
<td>35.5</td>
<td>51.0</td>
<td>293.3</td>
</tr>
<tr>
<td>moving target off</td>
<td>18/0</td>
<td>2</td>
<td>48.2</td>
<td>285.2</td>
<td>422.6</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>18/0</td>
<td>2</td>
<td>53.0</td>
<td>273.4</td>
<td>374.9</td>
</tr>
<tr>
<td><strong>Z3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>10/0</td>
<td>10</td>
<td>30.5</td>
<td>605.85</td>
<td>554.4</td>
</tr>
<tr>
<td>evasive off</td>
<td>20/0</td>
<td>0</td>
<td>35.5</td>
<td>64.2</td>
<td>388.35</td>
</tr>
<tr>
<td>moving target off</td>
<td>16/0</td>
<td>4</td>
<td>48.2</td>
<td>293</td>
<td>614.4</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>20/0</td>
<td>0</td>
<td>53.0</td>
<td>8.75</td>
<td>138.75</td>
</tr>
<tr>
<td><strong>Symba</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>10/0</td>
<td>10</td>
<td>30.5</td>
<td>654.5</td>
<td>632.9</td>
</tr>
<tr>
<td>evasive off</td>
<td>20/0</td>
<td>0</td>
<td>35.5</td>
<td>152.5</td>
<td>421.45</td>
</tr>
<tr>
<td>moving target off</td>
<td>12/0</td>
<td>8</td>
<td>48.2</td>
<td>632.05</td>
<td>644.4</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>20/0</td>
<td>0</td>
<td>53.0</td>
<td>118.1</td>
<td>187.85</td>
</tr>
</tbody>
</table>
## Experiments with WSN Benchmarks – 80-90% density

<table>
<thead>
<tr>
<th>Constraint settings</th>
<th>#SAT/UNSAT</th>
<th>#TO</th>
<th>Optimum</th>
<th>Runtime</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Puli’s linear search</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>20/0</td>
<td>0</td>
<td>119.3</td>
<td>93.4</td>
<td>184.3</td>
</tr>
<tr>
<td>evasive off</td>
<td>20/0</td>
<td>0</td>
<td>120.8</td>
<td>53.55</td>
<td>126.5</td>
</tr>
<tr>
<td>moving target off</td>
<td>20/0</td>
<td>0</td>
<td>117.6</td>
<td>141.15</td>
<td>135.95</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>20/0</td>
<td>0</td>
<td>116.8</td>
<td>40.55</td>
<td>75</td>
</tr>
<tr>
<td><strong>Puli’s binary search</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>20/0</td>
<td>0</td>
<td>119.3</td>
<td>42.3</td>
<td>176.1</td>
</tr>
<tr>
<td>evasive off</td>
<td>20/0</td>
<td>0</td>
<td>120.8</td>
<td>23.35</td>
<td>119.3</td>
</tr>
<tr>
<td>moving target off</td>
<td>20/0</td>
<td>0</td>
<td>117.6</td>
<td>35.8</td>
<td>127.9</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>20/0</td>
<td>0</td>
<td>116.8</td>
<td>11.5</td>
<td>67.55</td>
</tr>
<tr>
<td><strong>OptiMathSAT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>20/0</td>
<td>0</td>
<td>119.3</td>
<td>192.8</td>
<td>678.9</td>
</tr>
<tr>
<td>evasive off</td>
<td>19/0</td>
<td>1</td>
<td>120.8</td>
<td>190.5</td>
<td>457.9</td>
</tr>
<tr>
<td>moving target off</td>
<td>18/0</td>
<td>2</td>
<td>117.6</td>
<td>619.5</td>
<td>631.7</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>14/0</td>
<td>6</td>
<td>116.8</td>
<td>794.5</td>
<td>847.2</td>
</tr>
<tr>
<td><strong>Z3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>9/0</td>
<td>11</td>
<td>30.5</td>
<td>774.7</td>
<td>1198</td>
</tr>
<tr>
<td>evasive off</td>
<td>17/0</td>
<td>3</td>
<td>35.5</td>
<td>446.35</td>
<td>716.3</td>
</tr>
<tr>
<td>moving target off</td>
<td>12/0</td>
<td>8</td>
<td>48.2</td>
<td>589.55</td>
<td>699.35</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>20/0</td>
<td>0</td>
<td>53.0</td>
<td>12.65</td>
<td>126.15</td>
</tr>
<tr>
<td><strong>SYMBA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all on</td>
<td>4/0</td>
<td>16</td>
<td>30.5</td>
<td>1098.8</td>
<td>1237.9</td>
</tr>
<tr>
<td>evasive off</td>
<td>17/0</td>
<td>3</td>
<td>35.5</td>
<td>583.55</td>
<td>727.7</td>
</tr>
<tr>
<td>moving target off</td>
<td>11/0</td>
<td>9</td>
<td>48.2</td>
<td>820.2</td>
<td>916.8</td>
</tr>
<tr>
<td>evasive moving off</td>
<td>20/0</td>
<td>0</td>
<td>53.0</td>
<td>155.1</td>
<td>171.5</td>
</tr>
</tbody>
</table>
Another Example: the 0-1 Knapsack Problem

- **opt-resource-fun**: The remaining capacity in the knapsack.

\[
\text{wLimit} - \sum_{i=1}^{n} w_i x_i
\]

where

- \( n \geq 1 \): number of items to put in the knapsack
- \( w_i \) and \( v_i \): the weight and the value of the \( i^{th} \) item, respectively.
- \( \text{wLimit} \): the maximum weight capacity.
- \( x_i \): Boolean variable that denotes if the \( i^{th} \) item is put in the knapsack.

**Objective function**: \( \max : \sum_{i=1}^{n} v_i x_i \)

- **opt-resource-target**: 0

The Knapsack problem is non-monotonous.
Another Example: the 0-1 Knapsack Problem

- \text{opt-resource-fun} \Rightarrow \text{The remaining capacity in the knapsack.}

\[ wLimit - \sum_{i=1}^{n} w_i x_i \]

where

- \( n \geq 1 \): number of items to put in the knapsack
- \( w_i \) and \( v_i \): the weight and the value of the \( i^{th} \) item, respectively.
- \( wLimit \): the maximum weight capacity.
- \( x_i \): Boolean variable that denotes if the \( i^{th} \) item is put in the knapsack.

Objective function: \( \max : \sum_{i=1}^{n} v_i x_i \)

- \text{opt-resource-target} \Rightarrow 0

The Knapsack problem is non-monotonous.
150 items.

Values and weights are random numbers between 1 and 100.

Knapsack’s capacity is 70% of total weight.

<table>
<thead>
<tr>
<th>Solver</th>
<th>#SAT/UNSAT</th>
<th>#TO</th>
<th>Optimum</th>
<th>Runtime</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Puli’s binary search</td>
<td>30/0</td>
<td>0</td>
<td>255.1</td>
<td>3.3</td>
<td>45.93</td>
</tr>
<tr>
<td>Puli’s linear search</td>
<td>30/0</td>
<td>0</td>
<td>255.1</td>
<td>32.3</td>
<td>45.47</td>
</tr>
<tr>
<td>OptiMathSAT</td>
<td>30/0</td>
<td>0</td>
<td>255.1</td>
<td>103.7</td>
<td>110.73</td>
</tr>
</tbody>
</table>
Conclusion and Future Work

- New OMT solver **PULI** for QF_UFLIA.
- Seems to be really fast.
- You can boost the solving by providing a resource function.
- Ongoing work: **PULI** implemented in C as publicly available official release.
- Future work: Experimenting with binary search boosted by regression analysis.
- Future work: Now that we have an OMT solver scales enough, generate benchmarks for WSNs of larger size and of more complex model.
- Future work: Experimenting with further optimization problems.
- Future work: Experimenting with pseudo-Boolean and ILP solving.