Constrained Optimization Benchmark for Optimization Modulo Theories: a Cloud Resource Management Problem

Mădălina Erașcu[12] Răzvan Meteș[1]

[1]West University of Timișoara, Romania

[2]Institute e-Austria Timișoara, Romania

madalina.erascu@e-uvt.ro

July 7th, 2019

Outline

Motivation

Problem

Solution
- Solution Approaches
- Variables Encoding
- OMT Tools

Experimental Results

Discussion and Future Work

Contents

Motivation

Problem

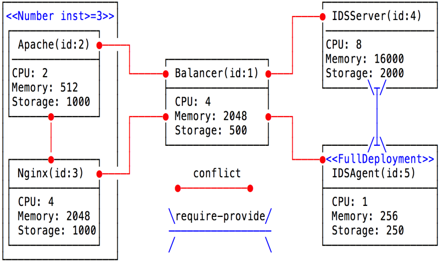
Solution
- Solution Approaches
- Variables Encoding
- OMT Tools

Experimental Results

Discussion and Future Work

Motivation

Problem: finding the best offer for a secure web container

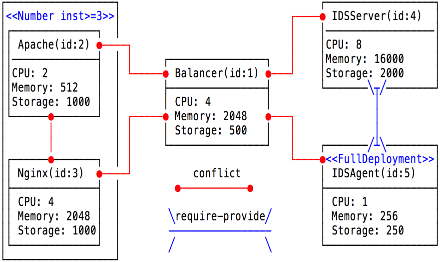


Components

- two Web Containers (e.g. Apache Tomcat or Nginx)
- a Balancer
- an IDSServer (Intrusion Detection System)
- an IDS Agent

Motivation

Problem: finding the best offer for a secure web container



Components

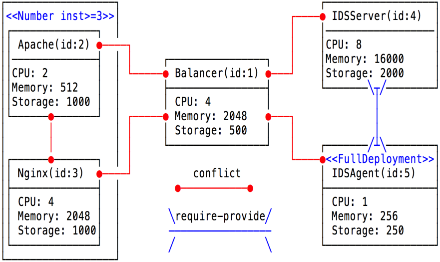
- two Web Containers (e.g. Apache Tomcat or Nginx)
- a Balancer
- an IDSServer (Intrusion Detection System)
- an IDS Agent

Constraints

- Conflicts: Balancer, Apache and Nginx cannot be deployed on the same VM
- Conflicts: Balancer and IDSServer needs exclusive use of machines
- Equal bound: exactly one Balancer has to be instantiated
- Lower bound: at least 3 instances of Apache and/or Nginx are required
- Require-provides: one IDSServer for 10 IDS Agents
- Full deployment: one instance of the IDS Agent on all VMs except for those containing the IDSServer and the Balancer
- Hardware constraints: components hardware requirements

Motivation

Problem: finding the best offer for a secure web container



Components

- two Web Containers (e.g. Apache Tomcat or Nginx)
- a Balancer
- an IDSServer (Intrusion Detection System)
- an IDS Agent

Constraints

- Conflicts: Balancer, Apache and Nginx cannot be deployed on the same VM
- Conflicts: Balancer and IDSServer needs exclusive use of machines
- Equal bound: exactly one Balancer has to be instantiated
- Lower bound: at least 3 instances of Apache and/or Nginx are required
- Require-provides: one IDSServer for 10 IDS Agents
- Full deployment: one instance of the IDS Agent on all VMs except for those containing the IDSServer and the Balancer
- Hardware constraints: components hardware requirements

Aim: find a set of virtual machines (VMs) which satisfy the components' requirements and lead to the minimum cost.

Cloud provider offers

Spot Instance Prices

Spot Instances | Defined Duration for Linux | Defined Duration for Windows

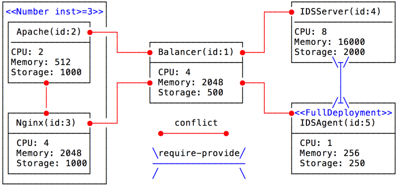
Region: EU (Ireland)

	Linux/UNIX Usage	Windows Usage
General Purpose - Current Generation		
t2.micro	$0.0038 per Hour	$0.0084 per Hour
t2.small	$0.0075 per Hour	$0.0165 per Hour
t2.medium	$0.015 per Hour	$0.033 per Hour
t2.large	$0.0302 per Hour	$0.0582 per Hour
t2.xlarge	$0.0605 per Hour	$0.1015 per Hour
t2.2xlarge	$0.121 per Hour	$0.183 per Hour
m3.medium	$0.0073 per Hour	$0.0633 per Hour
m3.large	$0.0306 per Hour	$0.1226 per Hour
m3.xlarge	$0.0612 per Hour	$0.2452 per Hour

Model	vCPU	CPU Credits / hour	Mem (GiB)	Storage
t2.nano	1	3	0.5	EBS-Only
t2.micro	1	6	1	EBS-Only
t2.small	1	12	2	EBS-Only
t2.medium	2	24	4	EBS-Only
t2.large	2	36	8	EBS-Only
t2.xlarge	4	54	16	EBS-Only
t2.2xlarge	8	81	32	EBS-Only

Remark: [snapshot from https://aws.amazon.com/ec2/] tens of thousands of price offers corresponding to different configurations and zones

Secure Web Container Use Case

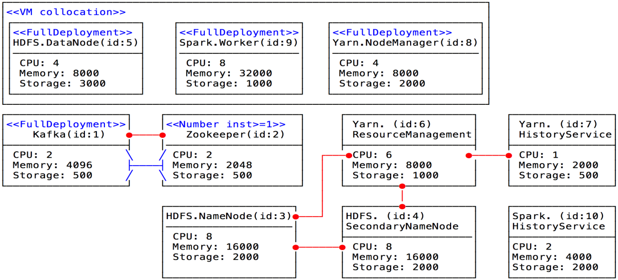


Example of a solution

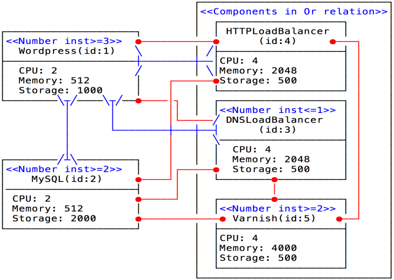
- VM_1 (CPU:8, RAM: 15 GB, Storage: 2000 GB, Price: 0.0526 $/hour): Nginx + IDS Agent
- VM_2 (CPU:4, RAM: 7.5 GB, Storage: 2000 GB, Price: 0.0283 $/hour): Balancer
- VM_3 (CPU:4, RAM: 30 GB, Storage: 2000 GB, Price: 0.0644 $/hour): IDSServer
- VM_4 (CPU:4, RAM: 7.5 GB, Storage: 2000 GB, Price: 0.0283 $/hour): Apache + IDS Agent
- VM_5 (CPU:4, RAM: 7.5 GB, Storage: 2000 GB, Price: 0.0283 $/hour): Apache + IDS Agent

Other use-cases considered

Oryx2 application



Wordpress application



Contents

Motivation

Problem

Solution
- Solution Approaches
- Variables Encoding
- OMT Tools

Experimental Results

Discussion and Future Work

Problem Formalization

Input

- A set of N components ($\{C_1, ..., C_N\}$) satisfying:

- A set of M VMs ($\{V_1, ..., V_M\}$)

Problem Formalization

Input

- A set of N components ($\{C_1, ..., C_N\}$) satisfying:
  - *Hardware constraints*
  - *Structural constraints*
- A set of M VMs ($\{V_1, ..., V_M\}$)

Problem Formalization

Input

- A set of N components ($\{C_1, ..., C_N\}$) satisfying:
  - *Hardware constraints*
  - *Structural constraints*
- A set of M VMs ($\{V_1, ..., V_M\}$)

Output

- A mapping a of components to VMs

$$a_{ik} = \begin{cases} 1 & \text{if } C_i \text{ is assigned to } V_k \\ 0 & \text{if } C_i \text{ is not assigned to } V_k \end{cases}$$

Problem Formalization

Input

- A set of N components ($\{C_1, ..., C_N\}$) satisfying:
  - *Hardware constraints*
  - *Structural constraints*
- A set of M VMs ($\{V_1, ..., V_M\}$)

Output

- A mapping a of components to VMs

$$a_{ik} = \begin{cases} 1 & \text{if } C_i \text{ is assigned to } V_k \\ 0 & \text{if } C_i \text{ is not assigned to } V_k \end{cases}$$

- which:
  - Satisfies the constraints induced by the interactions between components (structural constraints)
  - Satisfies the hardware requirements of all components (hardware constraints)
  - Minimizes the purchasing price

Problem Formalization (cont'd)

Default Constraints: $\sum_{i=1}^{N} a_{ik} \geq 1 \quad k = \overline{1, M}$

Problem Formalization (cont'd)

Default Constraints: $\sum_{i=1}^{N} a_{ik} \geq 1 \quad k = \overline{1, M}$

Hardware Constraints

- H^{res} – required amount of resource (CPU/Memory/Storage)
- $H^{res_{prov}}$ – the corresponding characteristic of a VM included in an existing Cloud Provider offer.

$$\sum_{i=1}^{N} a_{ik} H_i^{res} \leq H_k^{res_{prov}}, \quad k = \overline{1, M}$$

Problem Formalization (cont'd)

Default Constraints: $\sum_{i=1}^{N} a_{ik} \geq 1 \quad k = \overline{1, M}$

Hardware Constraints

- H^{res} – required amount of resource (CPU/Memory/Storage)
- $H^{res_{prov}}$ – the corresponding characteristic of a VM included in an existing Cloud Provider offer.

$$\sum_{i=1}^{N} a_{ik} H_i^{res} \leq H_k^{res_{prov}}, \quad k = \overline{1, M}$$

Structural Constraints

- Conflicts: two or more components cannot be deployed on the same VM

lin : $a_{ik} + a_{jk} \leq 1, k = \overline{1, M}, \forall (i, j) \text{s.t.} R_{ij} = 1$

nonlin : $a_{ik} a_{jk} \leq 1$

Problem Formalization (cont'd)

Default Constraints: $\sum_{i=1}^{N} a_{ik} \geq 1 \quad k = \overline{1, M}$

Hardware Constraints

- H^{res} – required amount of resource (CPU/Memory/Storage)
- $H^{res_{prov}}$ – the corresponding characteristic of a VM included in an existing Cloud Provider offer.

$$\sum_{i=1}^{N} a_{ik} H_i^{res} \leq H_k^{res_{prov}}, \quad k = \overline{1, M}$$

Structural Constraints

- Conflicts: two or more components cannot be deployed on the same VM

lin : $a_{ik} + a_{jk} \leq 1, k = \overline{1, M}, \forall (i, j) \text{s.t.} R_{ij} = 1$

nonlin : $a_{ik} a_{jk} \leq 1$

- Co-location: two or more components should be deployed on the same VM

$$a_{ik} = a_{jk}, \quad k = \overline{1, M}, \quad \forall (i, j) \text{ s.t. } D_{ij} = 1$$

Problem Formalization (cont'd)

Default Constraints: $\sum_{i=1}^{N} a_{ik} \geq 1 \quad k = \overline{1, M}$

Hardware Constraints

- H^{res} – required amount of resource (CPU/Memory/Storage)
- $H^{res_{prov}}$ – the corresponding characteristic of a VM included in an existing Cloud Provider offer.

$$\sum_{i=1}^{N} a_{ik} H_i^{res} \leq H_k^{res_{prov}}, \quad k = \overline{1, M}$$

Structural Constraints

- Conflicts: two or more components cannot be deployed on the same VM

$$\text{lin} : a_{ik} + a_{jk} \leq 1, k = \overline{1, M}, \forall (i, j) \text{s.t.} R_{ij} = 1$$

$$\text{nonlin} : a_{ik} a_{jk} \leq 1$$

- Co-location: two or more components should be deployed on the same VM

$$a_{ik} = a_{jk}, \quad k = \overline{1, M}, \quad \forall (i, j) \text{ s.t. } D_{ij} = 1$$

- Exclusive deployment: when from a set of q components only one should be deployed in a deployment plan

$$H(\sum_{k=1}^{M} a_{i_1 k}) + H(\sum_{k=1}^{M} a_{i_2 k}) + \ldots + H(\sum_{k=1}^{M} a_{i_q k}) = 1$$

$$H(u) = \begin{cases} 0 \text{ if } u = 0 \\ 1 \text{ if } u > 0 \end{cases}$$

Problem Formalization (cont'd)

Structural Constraints (cont'd)

- Require-provides 1: C_i requires (consumes) at least n_{ij} instances of C_j and C_j can serve (provides) at most m_{ij} instances of C_i.

$$n_{ij} \sum_{k=1}^{M} a_{ik} \leq m_{ij} \sum_{k=1}^{M} a_{jk}, \quad n_{ij}, m_{ij} \in \mathbb{N}.$$

Problem Formalization (cont'd)

Structural Constraints (cont'd)

- Require-provides 1: C_i requires (consumes) at least n_{ij} instances of C_j and C_j can serve (provides) at most m_{ij} instances of C_i.

$$n_{ij} \sum_{k=1}^{M} a_{ik} \leq m_{ij} \sum_{k=1}^{M} a_{jk}, \quad n_{ij}, m_{ij} \in \mathbb{N}.$$

- Require-provides 2: for each set of n instances of component C_j a new instance of C_i should be deployed

$$0 < n \sum_{k=1}^{M} a_{ik} - \sum_{k=1}^{M} a_{jk} \leq n, \quad n \in \mathbb{N}$$

Problem Formalization (cont'd)

Structural Constraints (cont'd)

- Require-provides 1: C_i requires (consumes) at least n_{ij} instances of C_j and C_j can serve (provides) at most m_{ij} instances of C_i.

$$n_{ij} \sum_{k=1}^{M} a_{ik} \leq m_{ij} \sum_{k=1}^{M} a_{jk}, \quad n_{ij}, m_{ij} \in \mathbb{N}.$$

- Require-provides 2: for each set of n instances of component C_j a new instance of C_i should be deployed

$$0 < n \sum_{k=1}^{M} a_{ik} - \sum_{k=1}^{M} a_{jk} \leq n, \quad n \in \mathbb{N}$$

- Full deployment: a component C_i must be deployed on all VMs (except on those which would induce conflicts)

$$\sum_{k=1}^{M} (a_{ik} + H(\sum_{j, R_{ij}=1} a_{jk})) = M$$

Problem Formalization (cont'd)

Structural Constraints (cont'd)

- Require-provides 1: C_i requires (consumes) at least n_{ij} instances of C_j and C_j can serve (provides) at most m_{ij} instances of C_i.

$$n_{ij} \sum_{k=1}^{M} a_{ik} \leq m_{ij} \sum_{k=1}^{M} a_{jk}, \quad n_{ij}, m_{ij} \in \mathbb{N}.$$

- Require-provides 2: for each set of n instances of component C_j a new instance of C_i should be deployed

$$0 < n \sum_{k=1}^{M} a_{ik} - \sum_{k=1}^{M} a_{jk} \leq n, \quad n \in \mathbb{N}$$

- Full deployment: a component C_i must be deployed on all VMs (except on those which would induce conflicts)

$$\sum_{k=1}^{M} (a_{ik} + H(\sum_{j, R_{ij}=1} a_{jk})) = M$$

- Deployment with bounded number of instances: the number of instances corresponding to a set of deployed components, $\overline{C}$, should be equal, greater or less than some values

$$\sum_{i \in \overline{C}} \sum_{k=1}^{M} a_{ik} \ \langle \mathrm{op} \rangle \ n,$$

$$\langle \mathrm{op} \rangle \in \{=, \leq, \geq\}, \ n \in \mathbb{N}$$

Problem Formalization (cont'd)

Constraints which encode the CPs offers and their link with the components hardware constraints

Problem Formalization (cont'd)

Constraints which encode the CPs offers and their link with the components hardware constraints

- $$\bigvee_{h=1}^{ON} vmType_k = h$$

where *vmType* identifies CPs offers ($vmType_k \in \{1, \ldots, ON\}$, where ON is the number of CP offers).

Problem Formalization (cont'd)

Constraints which encode the CPs offers and their link with the components hardware constraints

- $$\bigvee_{h=1}^{ON} vmType_k = h$$

where $vmType$ identifies CPs offers ($vmType_k \in \{1, \ldots, ON\}$, where ON is the number of CP offers).

- $$\sum_{i=1}^{N} a_{ik} \geq 1 \wedge vmType_k = h \implies VP_k = Price^{Offer_h} \wedge vm_k^{HR_1} = HR_1^{Offer_h} \wedge \ldots \wedge vm_k^{HR_L} = HR_L^{Offer_h}$$

Example: the first offer is e.g. $h = 1$ and $(Price^{Offer_1}, HR_1^{Offer_1}, HR_2^{Offer_1}, HR_3^{Offer_1})$ is (9.152$, 64, 488MB, 8GB)

Problem Formalization (cont'd)

Constraints which encode the CPs offers and their link with the components hardware constraints

- $$\bigvee_{h=1}^{ON} vmType_k = h$$

where *vmType* identifies CPs offers ($vmType_k \in \{1, \ldots, ON\}$, where ON is the number of CP offers).

- $$\sum_{i=1}^{N} a_{ik} \geq 1 \wedge vmType_k = h \implies VP_k = Price^{Offer_h} \wedge vm_k^{HR_1} = HR_1^{Offer_h} \wedge \ldots \wedge vm_k^{HR_L} = HR_L^{Offer_h}$$

Example: the first offer is e.g. $h = 1$ and $(Price^{Offer_1}, HR_1^{Offer_1}, HR_2^{Offer_1}, HR_3^{Offer_1})$ is (9.152\$, 64, 488MB, 8GB)

- $$\sum_{i=1}^{N} a_{ik} = 0 \implies VP_k = 0, \quad k = \overline{1, M}$$

Characteristics of the problem

- Constrained optimization

Characteristics of the problem

- Constrained optimization
- Linear programming: 0-1 + real/integer

Characteristics of the problem

- Constrained optimization
- Linear programming: 0-1 + real/integer
- Related to bin packing but ...

Characteristics of the problem

- Constrained optimization
- Linear programming: 0-1 + real/integer
- Related to bin packing but ...
  - ... the placement of items in bins is limited by constraints

Characteristics of the problem

- Constrained optimization
- Linear programming: 0-1 + real/integer
- Related to bin packing but ...
  - ... the placement of items in bins is limited by constraints
  - ... the capacity of bins is not fixed (it depends on the offers)

Characteristics of the problem

- Constrained optimization
- Linear programming: 0-1 + real/integer
- Related to bin packing but ...
  - ... the placement of items in bins is limited by constraints
  - ... the capacity of bins is not fixed (it depends on the offers)
  - ... the number of items is not known (it depends on the constraints on the number of instances)

Characteristics of the problem

- Constrained optimization
- Linear programming: 0-1 + real/integer
- Related to bin packing but ...
  - ... the placement of items in bins is limited by constraints
  - ... the capacity of bins is not fixed (it depends on the offers)
  - ... the number of items is not known (it depends on the constraints on the number of instances)
  - ... the smallest price is not necessarily obtained by using the smallest number of bins

Contents

Motivation

Problem

Solution
- Solution Approaches
- Variables Encoding
- OMT Tools

Experimental Results

Discussion and Future Work

Solution Approaches

1. Exact methods

Solution Approaches

1. Exact methods

2. Approximate methods

Solution Approaches

1. Exact methods
   - Constraint Programming (CP)[1]
     - Google or-tools (`https://github.com/google/or-tools`)
     - Advantages: can be interrupted after an amount of time giving the minimum found until that point
     - Drawback: large execution time

[1] F. Micota, M. Erașcu and D. Zaharie, "Constraint Satisfaction Approaches in Cloud Resource Selection for Component Based Applications," 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2018, pp. 443-450.

Solution Approaches

1. Exact methods

  - Constraint Programming (CP)

    - Google or-tools (`https://github.com/google/or-tools`)
    - Advantages: can be interrupted after an amount of time giving the minimum found until that point
    - Drawback: large execution time

  - Satisfiability/Optimization Modulo Theory (SMT/OMT) + symmetry breaking[2]

[2] M. Erașcu, F. Micota, and D. Zaharie. Influence of Variables Encoding and Symmetry Breaking on the Performance of Optimization Modulo Theories Tools Applied to Cloud Resource Selection. In G. Barthe, K. Korovin, S. Schulz, M. Suda, G. Sutcliffe, and M. Veanes, editors, LPAR-22 Workshop and Short Paper Proceedings, volume 9 of Kalpa Publications in Computing, pages 1–14. EasyChair, 2018.

Solution Approaches

1. Exact methods

   - Constraint Programming (CP)

     - Google or-tools (`https://github.com/google/or-tools`)
     - Advantages: can be interrupted after an amount of time giving the minimum found until that point
     - Drawback: large execution time

   - Satisfiability/Optimization Modulo Theory (SMT/OMT) + symmetry breaking

Solution Approaches

1. Exact methods
   - Constraint Programming (CP)
     - Google or-tools (`https://github.com/google/or-tools`)
     - Advantages: can be interrupted after an amount of time giving the minimum found until that point
     - Drawback: large execution time
   - Satisfiability/Optimization Modulo Theory (SMT/OMT) + symmetry breaking
2. Approximate methods
   - Population-based metaheuristic[1]
     - Evolutionary algorithm that uses only mutation operator
     - Advantages: always provides a (sub)optimal solution
     - Drawback: low success rate in case of larger instances

[1] F. Micota, M. Erașcu and D. Zaharie, "Constraint Satisfaction Approaches in Cloud Resource Selection for Component Based Applications," 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2018, pp. 443-450.

In this paper...

In this paper...

- proposal of basic benchmarking conventions in order to provide a thorough basis for reproducible and comparable benchmarking tests

In this paper...

- proposal of basic benchmarking conventions in order to provide a thorough basis for reproducible and comparable benchmarking tests
- testing of two state-of-the-art OMT solvers on different variable encodings and increasing problem dimension.

Benchmarking Conventions

Usage: provide a comprehensive benchmarking environment that allows to generate reproducible and comparable test results.

Benchmarking Conventions

Usage: provide a comprehensive benchmarking environment that allows to generate reproducible and comparable test results.

The developers of OMT intending to use the specified benchmark environment are advised to also give algorithmic details of their implementation.

Benchmarking Conventions

Usage: provide a comprehensive benchmarking environment that allows to generate reproducible and comparable test results.

The developers of OMT intending to use the specified benchmark environment are advised to also give algorithmic details of their implementation.

Conventions:

Benchmarking Conventions

Usage: provide a comprehensive benchmarking environment that allows to generate reproducible and comparable test results.

The developers of OMT intending to use the specified benchmark environment are advised to also give algorithmic details of their implementation.

Conventions:

- scaling the problem dimension:

Benchmarking Conventions

Usage: provide a comprehensive benchmarking environment that allows to generate reproducible and comparable test results.

The developers of OMT intending to use the specified benchmark environment are advised to also give algorithmic details of their implementation.

Conventions:

- scaling the problem dimension:
- usage of different variable encodings

Benchmarking Conventions

Usage: provide a comprehensive benchmarking environment that allows to generate reproducible and comparable test results.

The developers of OMT intending to use the specified benchmark environment are advised to also give algorithmic details of their implementation.

Conventions:

- scaling the problem dimension:
  - the number ON of VM offers as available on the Cloud Providers site; in our experiments we used $ON \in \{4; 10; 20; 40; 60; 80; 100\}$;
- usage of different variable encodings

Benchmarking Conventions

Usage: provide a comprehensive benchmarking environment that allows to generate reproducible and comparable test results.

The developers of OMT intending to use the specified benchmark environment are advised to also give algorithmic details of their implementation.

Conventions:

- scaling the problem dimension:
  - the number *ON* of VM offers as available on the Cloud Providers site; in our experiments we used $ON \in \{4; 10; 20; 40; 60; 80; 100\}$;
  - the number of components instances in constraints like *deployment with bounded number of instances* and *require-provides* to be deployed;
- usage of different variable encodings

Benchmarking Conventions

Usage: provide a comprehensive benchmarking environment that allows to generate reproducible and comparable test results.

The developers of OMT intending to use the specified benchmark environment are advised to also give algorithmic details of their implementation.

Conventions:

- scaling the problem dimension:
  - the number ON of VM offers as available on the Cloud Providers site; in our experiments we used $ON \in \{4; 10; 20; 40; 60; 80; 100\}$;
  - the number of components instances in constraints like *deployment with bounded number of instances* and *require-provides* to be deployed;
  - the number of hardware/software constraints; we have used CPU, memory and storage.
- usage of different variable encodings

Variables Encoding

Variable name	Type				
VP, HR_t, vm^{HR_t}, $vmType$	`Real`	`Int`	`BV`	`Real`	~~`Int`~~
a	`Real`	`Int`	`BV`	`Bool`	~~`Bool`~~

- All variables have type `Real` (`RealReal`).
  - Add explicitly the constraint $a_{ik} = 0 \vee a_{ik} = 1$

  where $i = \overline{1, N}$, $k = \overline{1, M}$

Variables Encoding

Variable name	Type				
VP, HR_t, vm^{HR_t}, $vmType$	`Real`	`Int`	`BV`	`Real`	~~`Int`~~
a	`Real`	`Int`	`BV`	`Bool`	~~`Bool`~~

- All variables have type `Int`.
  - *Case* 1: $a_{ik} = 0 \vee a_{ik} = 1$ (`IntIntOr`)
  - *Case* 2: $0 \leq a_{ik} \leq 1$ (`IntIntLessThan`)

  where $i = \overline{1, N}$, $k = \overline{1, M}$

Variables Encoding

Variable name	Type				
VP, HR_t, vm^{HR_t}, $vmType$	Real	Int	BV	Real	~~Int~~
a	Real	Int	BV	Bool	~~Bool~~

- All variables are bitvectors (BV).
  - bitvectors of size 32 (the constants involved are big integer numbers)
  - unsigned versions of the relational operators since the terms are only positive

Variables Encoding

Variable name	Type				
VP, HR_t, vm^{HR_t}, $vmType$	Real	Int	BV	Real	~~Int~~
a	Real	Int	BV	Bool	~~Bool~~

- All variables are `Real` except a which is `Bool`.
  - *Case* 1: Encode the hardware constraints using the `ite` operator to make a_{ik} and HR_t^i having compatible types (`RealBool`).

$$\sum_{i=1}^{N} a_{ik} \cdot HR_t^i \leq vm_k^{HR_t}, \quad vm_k^{HR_t}, HR_t^i \in \mathbb{R}_+, \quad k = \overline{1, M}, \quad t = \overline{1, L}$$

Variables Encoding

Variable name	Type				
VP, HR_t, vm^{HR_t}, $vmType$	Real	Int	BV	Real	~~Int~~
a	Real	Int	BV	Bool	~~Bool~~

- All variables are `Real` except a which is `Bool`.
  - *Case* 1: Encode the hardware constraints using the ite operator to make a_{ik} and HR_t^i having compatible types (`RealBool`).

$$\sum_{i=1}^{N} a_{ik} \cdot HR_t^i \leq vm_k^{HR_t}, \quad vm_k^{HR_t}, HR_t^i \in \mathbb{R}_+, \quad k = \overline{1, M}, \quad t = \overline{1, L}$$

  - *Case* 2: Encode some of the constraints as cardinality (e.g. default contraints) and pseudo-boolean constraints (e.g. hardware constraints)

$$\text{default} : \sum_{i=1}^{N} a_{ik} \geq 1 \quad k = \overline{1, M}$$

Variables Encoding

Variable name	Type				
VP, HR_t, vm^{HR_t}, $vmType$	Real	Int	BV	Real	~~Int~~
a	Real	Int	BV	Bool	~~Bool~~

- All variables are Real except a which is Bool.
  - *Case* 1: Encode the hardware constraints using the ite operator to make a_{ik} and HR_t^i having compatible types (RealBool).

$$\sum_{i=1}^{N} a_{ik} \cdot HR_t^i \leq vm_k^{HR_t}, \quad vm_k^{HR_t}, HR_t^i \in \mathbb{R}_+, \quad k = \overline{1, M}, \quad t = \overline{1, L}$$

  - *Case* 2: Encode some of the constraints as cardinality (e.g. default contraints) and pseudo-boolean constraints (e.g. hardware constraints)

    Remark 1: Not of all the constraints involved are cardinality/pseudo-boolean constraints so the problem is not an MaxSMT problem, hence a solver for QF_LRA/QF_NRA must be used.

$$\bigvee_{h=1}^{ON} vmType_k = h$$

$$\sum_{i=1}^{N} a_{ik} \geq 1 \wedge vmType_k = h \implies VP_k = Price^{Offer_h} \wedge vm_k^{HR_1} = HR_1^{Offer_h} \wedge \ldots \wedge vm_k^{HR_L} = HR_L^{Offer_h}$$

OMT Tools

Used the optimization modulo theory (OMT) tools:

OMT Tools

Used the optimization modulo theory (OMT) tools:

- OptiMathSAT (`http://optimathsat.disi.unitn.it`)

OMT Tools

Used the optimization modulo theory (OMT) tools:

- OptiMathSAT (`http://optimathsat.disi.unitn.it`)
- νZ (`https://github.com/Z3Prover/z3`)

OMT Tools

Used the optimization modulo theory (OMT) tools:

- OptiMathSAT (`http://optimathsat.disi.unitn.it`)
- νZ (`https://github.com/Z3Prover/z3`)

Remark 2:

- OptiMathSAT does not have support for the `RealPBC` encoding using the non-standard constructs like `atmost`, `atleast`, `exactly`, however they can be translated into `assert-soft` constraints (WiP).
- OptiMathSAT does not support non-linear constraints.
- In νZ, the cardinality constraints can be directly encoded, however the pseudo-boolean constraints can not (since a_{ik} is `Bool` and HR_t^i is `Real`).

OMT Tools

Used the optimization modulo theory (OMT) tools:

- OptiMathSAT (`http://optimathsat.disi.unitn.it`)
- νZ (`https://github.com/Z3Prover/z3`)

Remark 2:

- OptiMathSAT does not have support for the `RealPBC` encoding using the non-standard constructs like `atmost`, `atleast`, `exactly`, however they can be translated into `assert-soft` constraints (WiP).
- OptiMathSAT does not support non-linear constraints.
- In νZ, the cardinality constraints can be directly encoded, however the pseudo-boolean constraints can not (since a_{ik} is `Bool` and HR_t^i is `Real`).

Solution for νZ:

OMT Tools

Used the optimization modulo theory (OMT) tools:

- OptiMathSAT (`http://optimathsat.disi.unitn.it`)
- νZ (`https://github.com/Z3Prover/z3`)

Remark 2:

- OptiMathSAT does not have support for the `RealPBC` encoding using the non-standard constructs like `atmost`, `atleast`, `exactly`, however they can be translated into `assert-soft` constraints (WiP).
- OptiMathSAT does not support non-linear constraints.
- In νZ, the cardinality constraints can be directly encoded, however the pseudo-boolean constraints can not (since a_{ik} is `Bool` and HR_t^i is `Real`).

Solution for νZ:

- we used the ternary operator `ite`, transforming the type of a from `Bool` to `Real` in order to have compatible types for the variables involved in the constraints (`RealPBC`);

OMT Tools

Used the optimization modulo theory (OMT) tools:

- OptiMathSAT (`http://optimathsat.disi.unitn.it`)
- νZ (`https://github.com/Z3Prover/z3`)

Remark 2:

- OptiMathSAT does not have support for the `RealPBC` encoding using the non-standard constructs like `atmost`, `atleast`, `exactly`, however they can be translated into `assert-soft` constraints (WiP).
- OptiMathSAT does not support non-linear constraints.
- In νZ, the cardinality constraints can be directly encoded, however the pseudo-boolean constraints can not (since a_{ik} is `Bool` and HR_t^i is `Real`).

Solution for νZ:

- we used the ternary operator `ite`, transforming the type of a from `Bool` to `Real` in order to have compatible types for the variables involved in the constraints (`RealPBC`);
- we used the equivalent transformation of the hardware constraints (`RealPBCMultiObjectives`):

$$\sum_{i=1}^{N} a_{ik} \cdot HR_t^i \leq vm_k^{HR_t}, \quad vm_k^{HR_t}, HR_t^i \in \mathbb{R}_+, \quad k = \overline{1, M}, \quad t = \overline{1, L}$$

$$\rightsquigarrow$$

$$\neg a_{ik} \Rightarrow (a_{ik} H_i^{res} = 0) \ \wedge \ a_{ik} \Rightarrow (a_{ik} H_i^{res} = H_i^{res}) \wedge \texttt{minimize} \sum_{i=1}^{N} a_{ik} H_i^{res},$$

Contents

Motivation

Problem

Solution
- Solution Approaches
- Variables Encoding
- OMT Tools

Experimental Results

Discussion and Future Work

Experimental Results

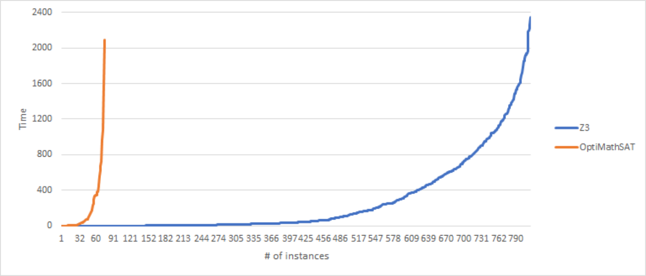


Figure: Comparison between νZ and OptiMathSAT

Experimental Results

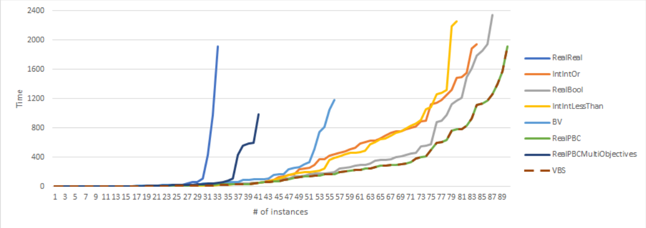


Figure: Scalability of νZ for different linear encodings

Experimental Results

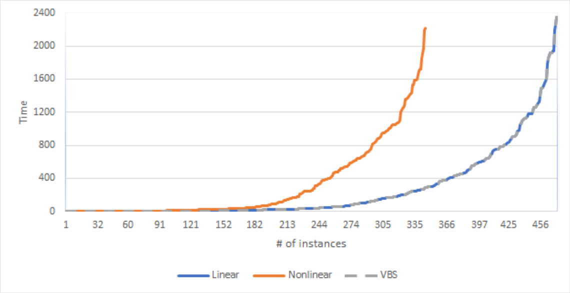


Figure: Comparison between linear/non-linear formalizations for νZ

Experimental Results

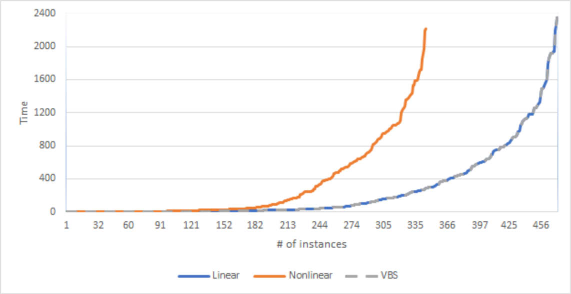


Figure: Comparison between linear/non-linear formalizations for νZ

- Scalability tests at: `https://github.com/merascu/Dissemination/tree/master/SMT2019`

Experimental Results

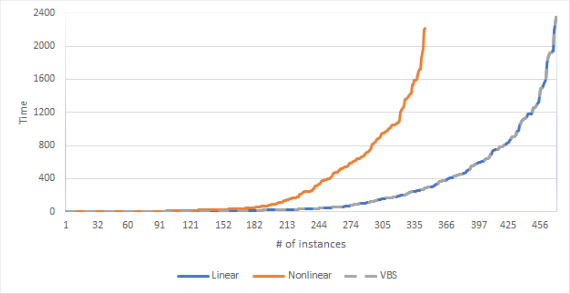


Figure: Comparison between linear/non-linear formalizations for νZ

- Scalability tests at: `https://github.com/merascu/Dissemination/tree/master/SMT2019`
- Benchmarks SMT-LIB-like at: `https://github.com/merascu/Optimization-Modulo-Theory/`

Contents

Motivation

Problem

Solution
- Solution Approaches
- Variables Encoding
- OMT Tools

Experimental Results

Discussion and Future Work

Discussion and Future Work

- It is not the number of VMs offers (number of variables and constraints), which decides the complexity of the problem, but rather the arithmetic of the constraints.

#offers=4		#offers =10		#offers =20		#offers =40		#offers =60		#offers =80		#offers =100	
νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSA
0.86	0.98	1.7	1.56	1.34	2.23	1.76	3.27	4.99	3.85	2.34	3.67	3.06	3.3

Table: Secure-Billing Email (BV encoding)

Discussion and Future Work

- It is not the number of VMs offers (number of variables and constraints), which decides the complexity of the problem, but rather the arithmetic of the constraints.

#offers=4		#offers =10		#offers =20		#offers =40		#offers =60		#offers =80		#offers =100	
νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSA
0.86	0.98	1.7	1.56	1.34	2.23	1.76	3.27	4.99	3.85	2.34	3.67	3.06	3.3

Table: Secure-Billing Email (BV encoding)

- In case of νZ, the timings at different runs can differ very much especially for constraints involving massive arithmetic; we can not comment if OptiMathSAT has similar behavior since it does not scale for such problems.

	#offers=4	#offers=10	#offers=20	#offers=40
Run 1	49.55	128.54	45.23	983.39
Run 2	59.85	120.37	133.11	2368.54
Run 3	62.33	45.38	280.34	2484.58
Run 4	58.71	81.65	125.51	1790.21
Run 5	44.31	131.43	105.61	1955.71
Average	54.95	101.47	137.96	1916.48

Table: Wordpress application: 4 instances of Wordpress to be deployed. The timings are for the RealPBC encoding. For more offers, the average timings is more than 40 minutes and are not listed.

Discussion and Future Work

- It is not the number of VMs offers (number of variables and constraints), which decides the complexity of the problem, but rather the arithmetic of the constraints.

#offers=4		#offers =10		#offers =20		#offers =40		#offers =60		#offers =80		#offers =100	
νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSAT	νZ	Opti MathSA
0.86	0.98	1.7	1.56	1.34	2.23	1.76	3.27	4.99	3.85	2.34	3.67	3.06	3.3

Table: Secure-Billing Email (`BV` encoding)

- In case of νZ, the timings at different runs can differ very much especially for constraints involving massive arithmetic; we can not comment if OptiMathSAT has similar behavior since it does not scale for such problems.

	#offers=4	#offers=10	#offers=20	#offers=40
Run 1	49.55	128.54	45.23	983.39
Run 2	59.85	120.37	133.11	2368.54
Run 3	62.33	45.38	280.34	2484.58
Run 4	58.71	81.65	125.51	1790.21
Run 5	44.31	131.43	105.61	1955.71
Average	54.95	101.47	137.96	1916.48

Table: Wordpress application: 4 instances of Wordpress to be deployed. The timings are for the `RealPBC` encoding. For more offers, the average timings is more than 40 minutes and are not listed.

- It seems that for problems not involving massive arithmetic, the `RealPBC` encoding gives the best results.

Discussion and Future Work

In order to improve the running time:

- Exploit the Boolean structure of the problem.

Discussion and Future Work

In order to improve the running time:

- Exploit the Boolean structure of the problem.
- Apply systematically symmetry breaking methods.