A symbiosis of constraint optimization, symmetries and symmetry breaking for scalable Cloud deployment problems

Mădălina Erașcu

West University of Timișoara, Romania

madalina.erascu@e-uvt.ro

Joint work with Bogdan David, Flavia Micota and Daniela Zaharie

The Tenth Congress of Romanian Mathematicians,
June 30 - July 5, 2023, Pitești, Romania

This work was patially supported by grants of the Romanian National Authority for Scientific Research and Innovation, CNCS/CCCDI - UEFISCDI: projects number PN-III-P2-2.1-PED-2016-0550 and PN-III-P1-1.1-TE-2021-0676 within PNCDI III.

Outline

Contents

Motivation

Advent of Cloud computing $\rightsquigarrow$

Motivation

Advent of Cloud computing $\rightsquigarrow$ loosely-coupled architecture

Motivation

Advent of Cloud computing $\rightsquigarrow$ loosely-coupled architecture $\rightsquigarrow$ DevOps paradigm

Motivation

Advent of Cloud computing $\rightsquigarrow$ loosely-coupled architecture $\rightsquigarrow$ DevOps paradigm $\rightsquigarrow$ application modeling

Motivation

Advent of Cloud computing $\rightsquigarrow$ loosely-coupled architecture $\rightsquigarrow$ DevOps paradigm $\rightsquigarrow$ application modeling $\rightsquigarrow$ optimal deployment

Motivation

Advent of Cloud computing $\rightsquigarrow$ loosely-coupled architecture $\rightsquigarrow$ DevOps paradigm $\rightsquigarrow$ application modeling $\rightsquigarrow$ optimal deployment

Benefits of optimal deployment:

Motivation

Advent of Cloud computing $\rightsquigarrow$ loosely-coupled architecture $\rightsquigarrow$ DevOps paradigm $\rightsquigarrow$ application modeling $\rightsquigarrow$ optimal deployment

Benefits of optimal deployment:

1. the synthesis of deployment plans that are optimal by design

Motivation

Advent of Cloud computing $\rightsquigarrow$ loosely-coupled architecture $\rightsquigarrow$ DevOps paradigm $\rightsquigarrow$ application modeling $\rightsquigarrow$ optimal deployment

Benefits of optimal deployment:

1. the synthesis of deployment plans that are optimal by design
2. the integration of such deployment plans into the application modeling process, enables formal reasoning on a model of the deployed application.

Contents

Problem Specification

Automated deployment of component-based applications in the Cloud consists of:

Problem Specification

Automated deployment of component-based applications in the Cloud consists of:

1. selection of the computing resources,

Problem Specification

Automated deployment of component-based applications in the Cloud consists of:

1. selection of the computing resources,
2. distribution/assignment of the application components over the available computing resources,

Problem Specification

Automated deployment of component-based applications in the Cloud consists of:

1. selection of the computing resources,
2. distribution/assignment of the application components over the available computing resources,
3. its dynamic modification to cope with peaks of user requests.

Problem Specification

Automated deployment of component-based applications in the Cloud consists of:

1. selection of the computing resources,
2. distribution/assignment of the application components over the available computing resources,

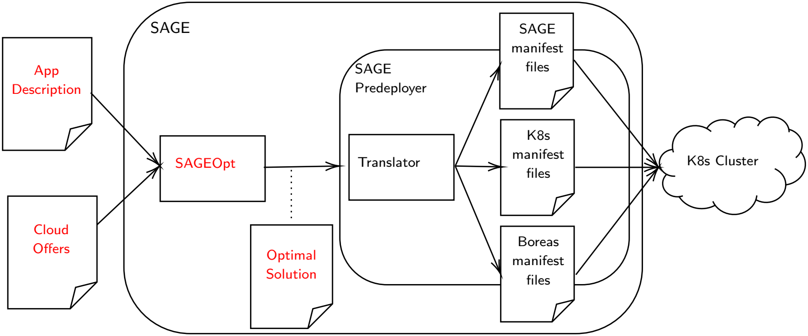


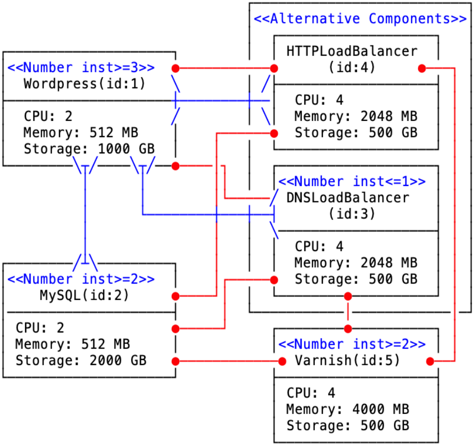
Figure: SAGE General Architecture

Contents

Contents

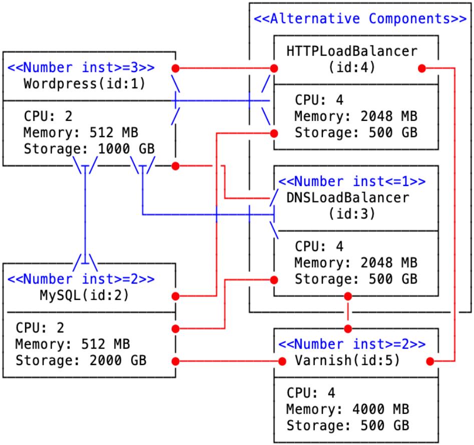
Case Study: Wordpress: Formalization

Wordpress (`www.wordpress.com`) is an open-source application frequently used in creating websites, blogs and web applications.



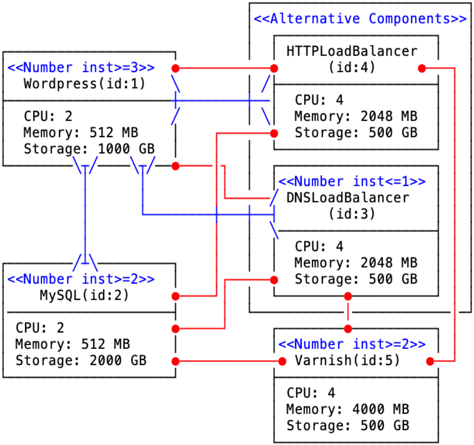
Case Study: Wordpress: Formalization

Wordpress (`www.wordpress.com`) is an open-source application frequently used in creating websites, blogs and web applications.



Case Study: Wordpress: Formalization

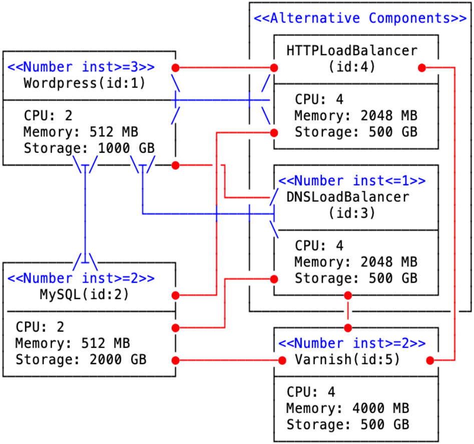
Wordpress (www.wordpress.com) is an open-source application frequently used in creating websites, blogs and web applications.



- DNSLoadBalancer requires at least 1 instance of Wordpress and can serve at most 7 such instances (*Require-Provide constraint*)

Case Study: Wordpress: Formalization

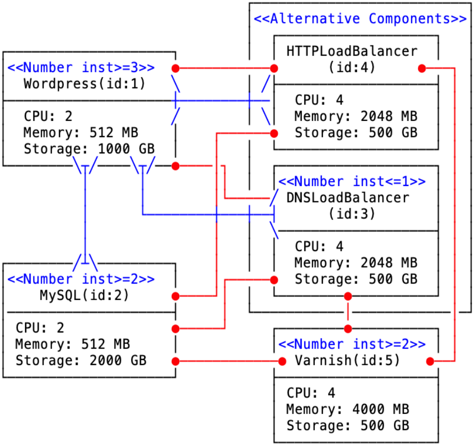
Wordpress (`www.wordpress.com`) is an open-source application frequently used in creating websites, blogs and web applications.



- DNSLoadBalancer requires at least 1 instance of Wordpress and can serve at most 7 such instances (*Require-Provide constraint*)
- Only one type of balancer must be deployed (*Exclusive deployment constraint*).

Case Study: Wordpress: Formalization

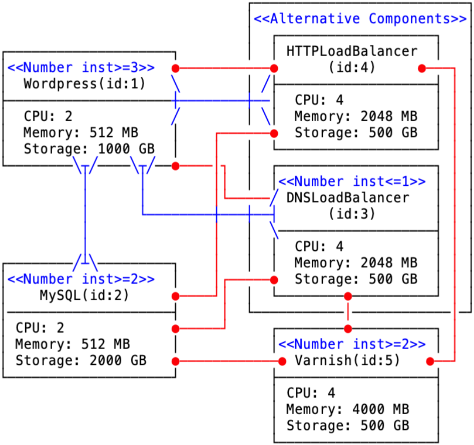
Wordpress (`www.wordpress.com`) is an open-source application frequently used in creating websites, blogs and web applications.



- DNSLoadBalancer requires at least 1 instance of Wordpress and can serve at most 7 such instances (*Require-Provide constraint*)
- Only one type of balancer must be deployed (*Exclusive deployment constraint*).
- Components are characterized in terms of their resource demand (i.e. in terms of CPU cores, RAM and storage capacity).

Case Study: Wordpress: Formalization

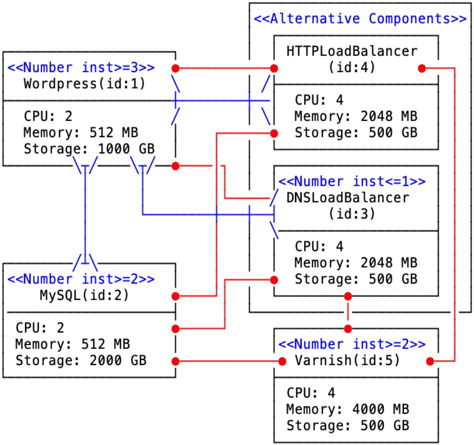
Wordpress (`www.wordpress.com`) is an open-source application frequently used in creating websites, blogs and web applications.



- DNSLoadBalancer requires at least 1 instance of Wordpress and can serve at most 7 such instances (*Require-Provide constraint*)
- Only one type of balancer must be deployed (*Exclusive deployment constraint*).
- Components are characterized in terms of their resource demand (i.e. in terms of CPU cores, RAM and storage capacity).
- ...

Case Study: Wordpress: Formalization

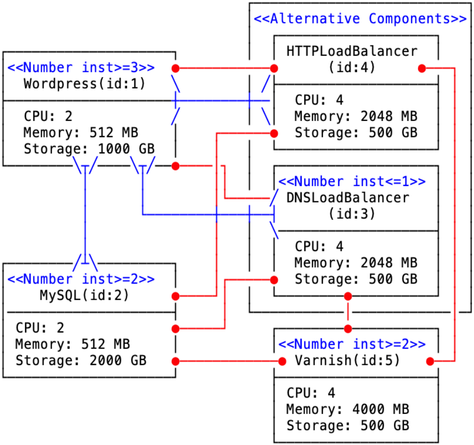
Wordpress (`www.wordpress.com`) is an open-source application frequently used in creating websites, blogs and web applications.



- DNSLoadBalancer requires at least 1 instance of Wordpress and can serve at most 7 such instances (*Require-Provide constraint*) $\sum_{k=1}^{M} a_{3k} \leq 7 \sum_{k=1}^{M} a_{1k}, \;\; k = \overline{1, M}$
- Only one type of balancer must be deployed (*Exclusive deployment constraint*).
- Components are characterized in terms of their resource demand (i.e. in terms of CPU cores, RAM and storage capacity).
- ...

Case Study: Wordpress: Formalization

Wordpress (`www.wordpress.com`) is an open-source application frequently used in creating websites, blogs and web applications.



- DNSLoadBalancer requires at least 1 instance of Wordpress and can serve at most 7 such instances (*Require-Provide constraint*)
  $\sum_{k=1}^{M} a_{3k} \leq 7 \sum_{k=1}^{M} a_{1k}, \ \ k = \overline{1, M}$
- Only one type of balancer must be deployed (*Exclusive deployment constraint*).
  $a_{3k} + a_{4k} \ \leq \ 1, \ \ k = \overline{1, M}$
- Components are characterized in terms of their resource demand (i.e. in terms of CPU cores, RAM and storage capacity).
- ...

Cloud provider offers

Spot Instance Prices

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

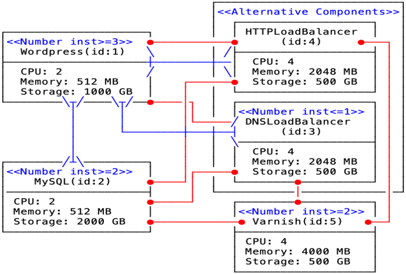
Region: EU (Ireland)

	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

Wordpress: Example Solution



- VM_1 (CPU:4, RAM: 30.5 GB, Storage: 1000 GB, Price: 0.0379 \$/hour): Wordpress+MySQL
- VM_2 (CPU:4, RAM: 30.5 GB, Storage: 1000 GB, Price: 0.0379 \$/hour): Wordpress+MySQL
- VM_3 (CPU:4, RAM: 7.5 GB, Storage: 2000 GB, Price: 0.021 \$/hour): Varnish
- VM_4 (CPU:4, RAM: 7.5 GB, Storage: 2000 GB, Price: 0.021 \$/hour): Varnish
- VM_5 (CPU:4, RAM: 7.5 GB, Storage: 2000 GB, Price: 0.021 \$/hour): HTTPLoadBalancer
- VM_6 (CPU:4, RAM: 7.5 GB, Storage: 2000 GB, Price: 0.021 \$/hour): Wordpress

Contents

Problem Formalization

General constraints

$$\begin{array}{rll}
\textit{Basic allocation} & \sum_{k=1}^{M} a_{ik} \geq 1 & \forall i = \overline{1, N} \\
\textit{Occupancy} & \sum_{i=1}^{N} a_{ik} \geq 1 \Rightarrow v_k = 1 & \forall k = \overline{1, M} \\
\textit{Capacity} & \sum_{i=1}^{N} a_{ik} \cdot R_i^h \leq F_{t_k}^h & \forall k = \overline{1, M}, \forall h = \overline{1, H} \\
\textit{Link} & v_k{=}1 \wedge t_k{=}o \Rightarrow \bigwedge_{h=1}^{H} \left(r_k^h{=}F_{t_k}^h\right) \wedge p_k{=}P_{t_k} & \forall o = \overline{1, O},\ O \in \mathbb{N}^* \\
 & \sum_{i=1}^{N} a_{ik} = 0 \Rightarrow t_k = 0 & \forall k = \overline{1, M}
\end{array}$$

where:

- $R_i^h \in \mathbb{N}^*$ is the hardware requirement of type h of the component i;
- $F_{t_k}^h \in \mathbb{N}^*$ is the hardware characteristic h of the VM of type t_k.

Problem Formalization (cont'd)

Application-specific constraints

Conflicts	$a_{ik} + a_{jk} \leq 1$	$\forall k = \overline{1, M},\ \forall (i,j)\ \mathcal{R}_{ij} = 1$
Co-location	$a_{ik} = a_{jk}$	$\forall k = \overline{1, M},\ \forall (i,j)\ \mathcal{D}_{ij} = 1$
Exclusive	*deployment*	
	$\mathcal{H}\left(\sum_{k=1}^{M} a_{i_1 k}\right) + \ldots + \mathcal{H}\left(\sum_{k=1}^{M} a_{i_q k}\right) = 1$	for fixed $q \in \{1, \ldots, N\}$
		$\mathcal{H}(u) = \begin{cases} 1 & u > 0 \\ 0 & u = 0 \end{cases}$
Require-	*Provide*	
	$n_{ij} \sum_{k=1}^{M} a_{ik} \leq m_{ij} \sum_{k=1}^{M} a_{jk}$	$\forall (i,j) \mathcal{Q}_{ij}(n_{ij}, m_{ij}) = 1$
	$0 \leq n \sum_{k=1}^{M} a_{jk} - \sum_{k=1}^{M} a_{ik} < n$	$n, n_{ij}, m_{ij} \in \mathbb{N}^*$

where:

- $\mathcal{R}_{ij} = 1$ if components i and j are in conflict (can not be placed in the same VM);
- $\mathcal{D}_{ij} = 1$ if components i and j must be co-located (must be placed in the same VM);
- $\mathcal{Q}_{ij}(n, m)$=1 if C_i requires at least n instances of C_j and C_j can serve at most m instances of C_i

Problem Formalization (cont'd)

Application-specific constraints

Full deployment $\sum_{k=1}^{M}\left(a_{ik} + \mathcal{H}\left(\sum_{j,\mathcal{R}_{ij}=1} a_{jk}\right)\right) = \sum_{k=1}^{M} v_k$

Deployment with bounded number of instances

$\sum_{i\in\overline{C}} \sum_{k=1}^{M} a_{ik}\langle\text{op}\rangle n$ $\quad |\overline{C}| \leq N,\ \langle\text{op}\rangle \in \{=,\leq,\geq\},\ n\in\mathbb{N}$

Find:

- assignment matrix a with binary entries $a_{ik} \in \{0,1\}$ for $i = \overline{1,N}$, $k = \overline{1,M}$, which are interpreted as follows:

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

- the type selection vector t with integer entries t_k for $k = \overline{1,M}$, representing the type (from a predefined set) of each VM leased.

Such that: the leasing price is minimal $\sum_{k=1}^{M} v_k \cdot p_k$

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

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
- NP-hard

Contents

Solution Approaches

1. Exact methods

Solution Approaches

1. Exact methods

2. Approximate methods

Solution Approaches

1. Exact methods
   - Constrained Programming (CP)* °
     - Modelling language: MiniZinc (`https://www.minizinc.org`)
     - Solvers integrated with MiniZinc: Google OR-Tools, Gecode, Chuffed

° B. David, "Constraint Optimization Approaches for Cloud Resource Provisioning," National Scientific Session of Mathematics and Informatics, November 25-27, 2021, Brasov, Romania.
* 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.
** M Erașcu, F Micota, D Zaharie, "Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking", Journal of Logical and Algebraic Methods in Programming 121, 100664

Solution Approaches

1. Exact methods
   - Constrained Programming (CP)* °
     - Modelling language: MiniZinc (`https://www.minizinc.org`)
     - Solvers integrated with MiniZinc: Google OR-Tools, Gecode, Chuffed
   - Mathematical Programming (MP)**
     - Python CPLEX API

° B. David, "Constraint Optimization Approaches for Cloud Resource Provisioning," National Scientific Session of Mathematics and Informatics, November 25-27, 2021, Brasov, Romania.

* 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.

** M Erașcu, F Micota, D Zaharie, "Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking", Journal of Logical and Algebraic Methods in Programming 121, 100664

Solution Approaches

1. Exact methods
   - Constrained Programming (CP)* °
     - Modelling language: MiniZinc (https://www.minizinc.org)
     - Solvers integrated with MiniZinc: Google OR-Tools, Gecode, Chuffed
   - Mathematical Programming (MP)**
     - Python CPLEX API
   - Satisfiability Modulo Theory (SMT)**
     - Python Z3 API

° B. David, "Constraint Optimization Approaches for Cloud Resource Provisioning," National Scientific Session of Mathematics and Informatics, November 25-27, 2021, Brasov, Romania.
* 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.
** M Erașcu, F Micota, D Zaharie, "Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking", Journal of Logical and Algebraic Methods in Programming 121, 100664

Solution Approaches

1. Exact methods
   - Constrained Programming (CP)* °
     - Modelling language: MiniZinc (`https://www.minizinc.org`)
     - Solvers integrated with MiniZinc: Google OR-Tools, Gecode, Chuffed
   - Mathematical Programming (MP)**
     - Python CPLEX API
   - Satisfiability Modulo Theory (SMT)**
     - Python Z3 API
   - Advantage: provides an optimal solution

° B. David, "Constraint Optimization Approaches for Cloud Resource Provisioning," National Scientific Session of Mathematics and Informatics, November 25-27, 2021, Brasov, Romania.
* 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.
** M Erașcu, F Micota, D Zaharie, "Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking", Journal of Logical and Algebraic Methods in Programming 121, 100664

Solution Approaches

1. Exact methods
   - Constrained Programming (CP)* °
     - Modelling language: MiniZinc (`https://www.minizinc.org`)
     - Solvers integrated with MiniZinc: Google OR-Tools, Gecode, Chuffed
   - Mathematical Programming (MP)**
     - Python CPLEX API
   - Satisfiability Modulo Theory (SMT)**
     - Python Z3 API
   - Advantage: provides an optimal solution
   - Drawback: significant computational time for large problems

° B. David, "Constraint Optimization Approaches for Cloud Resource Provisioning," National Scientific Session of Mathematics and Informatics, November 25-27, 2021, Brasov, Romania.
* 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.
** M Erașcu, F Micota, D Zaharie, "Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking", Journal of Logical and Algebraic Methods in Programming 121, 100664

Solution Approaches

1. Exact methods
   - Constrained Programming (CP)* °
     - Modelling language: MiniZinc (`https://www.minizinc.org`)
     - Solvers integrated with MiniZinc: Google OR-Tools, Gecode, Chuffed
   - Mathematical Programming (MP)**
     - Python CPLEX API
   - Satisfiability Modulo Theory (SMT)**
     - Python Z3 API
   - Advantage: provides an optimal solution
   - Drawback: significant computational time for large problems
2. Approximate methods
   - Population-based metaheuristic*

° B. David, "Constraint Optimization Approaches for Cloud Resource Provisioning," National Scientific Session of Mathematics and Informatics, November 25-27, 2021, Brasov, Romania.
* 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.
** M Erașcu, F Micota, D Zaharie, "Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking", Journal of Logical and Algebraic Methods in Programming 121, 100664

Solution Approaches

1. Exact methods
   - Constrained Programming (CP)* °
     - Modelling language: MiniZinc (`https://www.minizinc.org`)
     - Solvers integrated with MiniZinc: Google OR-Tools, Gecode, Chuffed
   - Mathematical Programming (MP)**
     - Python CPLEX API
   - Satisfiability Modulo Theory (SMT)**
     - Python Z3 API
   - Advantage: provides an optimal solution
   - Drawback: significant computational time for large problems
2. Approximate methods
   - Population-based metaheuristic*
     - Evolutionary algorithm that uses only mutation operator

° B. David, "Constraint Optimization Approaches for Cloud Resource Provisioning," National Scientific Session of Mathematics and Informatics, November 25-27, 2021, Brasov, Romania.
* 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.
** M Erașcu, F Micota, D Zaharie, "Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking", Journal of Logical and Algebraic Methods in Programming 121, 100664

Solution Approaches

1. Exact methods
   - Constrained Programming (CP)* °
     - Modelling language: MiniZinc (`https://www.minizinc.org`)
     - Solvers integrated with MiniZinc: Google OR-Tools, Gecode, Chuffed
   - Mathematical Programming (MP)**
     - Python CPLEX API
   - Satisfiability Modulo Theory (SMT)**
     - Python Z3 API
   - Advantage: provides an optimal solution
   - Drawback: significant computational time for large problems
2. Approximate methods
   - Population-based metaheuristic*
     - Evolutionary algorithm that uses only mutation operator
     - Advantage: always provides a (sub)optimal solution

° B. David, "Constraint Optimization Approaches for Cloud Resource Provisioning," National Scientific Session of Mathematics and Informatics, November 25-27, 2021, Brasov, Romania.
* 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.
** M Erașcu, F Micota, D Zaharie, "Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking", Journal of Logical and Algebraic Methods in Programming 121, 100664

Solution Approaches

1. Exact methods
   - Constrained Programming (CP)* °
     - Modelling language: MiniZinc (https://www.minizinc.org)
     - Solvers integrated with MiniZinc: Google OR-Tools, Gecode, Chuffed
   - Mathematical Programming (MP)**
     - Python CPLEX API
   - Satisfiability Modulo Theory (SMT)**
     - Python Z3 API
   - Advantage: provides an optimal solution
   - Drawback: significant computational time for large problems
2. Approximate methods
   - Population-based metaheuristic*
     - Evolutionary algorithm that uses only mutation operator
     - Advantage: always provides a (sub)optimal solution
     - Drawback: low success rate in case of larger instances

° B. David, "Constraint Optimization Approaches for Cloud Resource Provisioning," National Scientific Session of Mathematics and Informatics, November 25-27, 2021, Brasov, Romania.
* 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.
** M Erașcu, F Micota, D Zaharie, "Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking", Journal of Logical and Algebraic Methods in Programming 121, 100664

In this presentation: Speeding-up exact methods by symmetry breaking

Case Study: Wordpress (cont'd)

Solution:

- *assignment matrix* with elements $a_{ij} \in \{0, 1\}$

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

- *type selection vector t* with elements $t_k \in \mathbb{N}$ ($k = \overline{1, M}$) representing the type (from a predefined set) of each VM leased.

both fulfilling the application constraints and minimizing the leasing price.

Case Study: Wordpress (cont'd)

Solution:

- *assignment matrix* with elements $a_{ij} \in \{0, 1\}$

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

- *type selection vector t* with elements $t_k \in \mathbb{N}$ $(k = \overline{1, M})$ representing the type (from a predefined set) of each VM leased.

both fulfilling the application constraints and minimizing the leasing price.

For the case when the number of Wordpress instances is 3, we have:

- the assignment matrix

	V_1	V_2	V_3	V_4	V_5	V_6
$Wordpress(C_1)$	1	1	0	0	0	1
$MySql(C_2)$	1	1	0	0	0	0
$DNSLoadBalancer(C_3)$	0	0	0	0	0	0
$HTTPLoadBalancer(C_4)$	0	0	0	0	1	0
$Varnish(C_5)$	0	0	1	1	0	0

- the type vector: $t = [186, 186, 182, 182, 182, 182]$.

Contents

Experimental Analysis I

Goals:

Experimental Analysis I

Goals:

- study the scalability of state-of-the-art general CP, MP and SMT tools in solving COPs corresponding to the deployment of component-based applications in the Cloud

Experimental Analysis I

Goals:

- study the scalability of state-of-the-art general CP, MP and SMT tools in solving COPs corresponding to the deployment of component-based applications in the Cloud
- effectiveness of various static symmetry breaking techniques in improving the computational time of solving these problems.

Experimental Analysis I

Goals:

- study the scalability of state-of-the-art general CP, MP and SMT tools in solving COPs corresponding to the deployment of component-based applications in the Cloud
- effectiveness of various static symmetry breaking techniques in improving the computational time of solving these problems.
- the evaluation from two perspectives: number of VMs offers, respectively number of deployed instances of components.

Experimental Analysis I

Goals:

- study the scalability of state-of-the-art general CP, MP and SMT tools in solving COPs corresponding to the deployment of component-based applications in the Cloud
- effectiveness of various static symmetry breaking techniques in improving the computational time of solving these problems.
- the evaluation from two perspectives: number of VMs offers, respectively number of deployed instances of components.

#ins	#ofr=20	#ofr=40	#ofr=250	#ofr=500
		Chuffed		
3	2.13	4.18	57.72	458.21
4	26.03	114.18	1974.16	-
5	638.26	2230.39	-	-
		OR-Tools		
3	3.52	8.42	96.38	191.38
4	23.25	56.47	502.71	988.33
5	149.47	428.98	-	-
6	494.46	1174.36	-	-
		IBM CPLEX		
3	9.81	-	-	-
4	124.68	-	-	-
5	452.32	-	-	-
6	737.89	-	-	-
		Z3		
3	2.92	4.13	115.36	391.87
4	46.46	366.24	-	-

Symmetries

Symmetries

- A symmetry is a bijection on decision variables (i.e. a, t) that preserves solutions and non-solutions.

Symmetries

- A symmetry is a bijection on decision variables (i.e. a, t) that preserves solutions and non-solutions.
- Symmetry often occurs because groups of objects within a matrix are indistinguishable. This leads to row/column symmetries.

Symmetries

- A symmetry is a bijection on decision variables (i.e. a, t) that preserves solutions and non-solutions.
- Symmetry often occurs because groups of objects within a matrix are indistinguishable. This leads to row/column symmetries.
- Two variables are indistinguishable if some symmetry interchanges their roles in all solutions and non-solutions (variable symmetry).

Symmetries

- A symmetry is a bijection on decision variables (i.e. a, t) that preserves solutions and non-solutions.
- Symmetry often occurs because groups of objects within a matrix are indistinguishable. This leads to row/column symmetries.
- Two variables are indistinguishable if some symmetry interchanges their roles in all solutions and non-solutions (variable symmetry).
- Value symmetries can permute only the values of variables.

Symmetries

- A symmetry is a bijection on decision variables (i.e. a, t) that preserves solutions and non-solutions.
- Symmetry often occurs because groups of objects within a matrix are indistinguishable. This leads to row/column symmetries.
- Two variables are indistinguishable if some symmetry interchanges their roles in all solutions and non-solutions (variable symmetry).
- Value symmetries can permute only the values of variables.
- A matrix has row/column symmetry iff all the rows/columns of one of its matrices are indistinguishable.

Symmetries

- A symmetry is a bijection on decision variables (i.e. a, t) that preserves solutions and non-solutions.
- Symmetry often occurs because groups of objects within a matrix are indistinguishable. This leads to row/column symmetries.
- Two variables are indistinguishable if some symmetry interchanges their roles in all solutions and non-solutions (variable symmetry).
- Value symmetries can permute only the values of variables.
- A matrix has row/column symmetry iff all the rows/columns of one of its matrices are indistinguishable.
- A matrix has partial row/column symmetry iff strict subset(s) of the rows/columns are indistinguishable.

Symmetries

- A symmetry is a bijection on decision variables (i.e. a, t) that preserves solutions and non-solutions.
- Symmetry often occurs because groups of objects within a matrix are indistinguishable. This leads to row/column symmetries.
- Two variables are indistinguishable if some symmetry interchanges their roles in all solutions and non-solutions (variable symmetry).
- Value symmetries can permute only the values of variables.
- A matrix has row/column symmetry iff all the rows/columns of one of its matrices are indistinguishable.
- A matrix has partial row/column symmetry iff strict subset(s) of the rows/columns are indistinguishable.

Partial row/column symmetry are more often encountered in Cloud deployment problems.

Symmetry Breaking: Column Symmetries

Ordering decreasing

- (L) the columns by the number of components for columns representing VMs of the same type:

$$\sum_{i=1}^{N} a_{ik} \geq \sum_{i=1}^{N} a_{i(k+1)}, \quad \forall k = \overline{1, N-1}$$

- (LX) the columns by lexicographic order for columns representing VMs of the same type

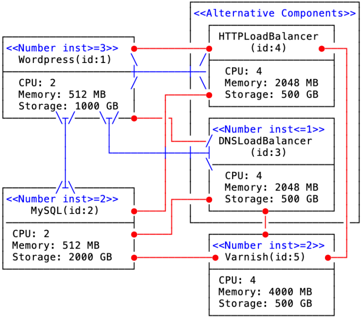
$$a_{\star k} \succ_{lex} a_{\star(k+1)}, \text{ where } a_{\star k} \text{ denotes the column } k.$$

- (PR) ordering decreasing the VMs by their characteristics (price, CPU, memory, storage)

$$P_1 \geq P_2 \geq ... \geq P_N, \quad \forall k = \overline{1, N}$$

Symmetry Breaking: Row Symmetries

(FV) pre-assigning, on separate VMs, the components composing the clique with maximum deployment size obtained from the conflict graph, i.e. the graph where the component instances are the nodes and the conflicts are the edges.

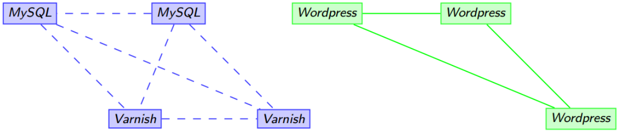


Example (**FV**: Wordpress with 3 Wordpress instances)

There are 3 cliques with maximum deployment size 4. Pick one:

- $[2 MySQL, 2 Varnish]$
- ~~$[3 Wordpress, 1 HTTPLoadBalancer]$~~
- ~~$[3 Wordpress, 1 DNSLoadBalancer]$~~

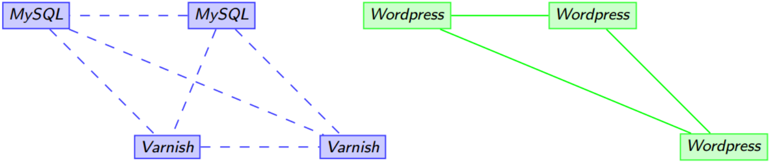
Examples of cliques



Symmetry Breaking: Row Symmetries (cont'd)

Example (**FV**: Wordpress with 3 Wordpress instances)

Clique with maximum deployment size 4: $[2MySQL, 2Varnish]$



	V_1	V_2	V_3	V_4	V_5	V_6
C_1	?	?	?	?	?	?
C_2	1	1	0	0	?	?
C_3	?	?	?	?	?	?
C_4	?	?	?	?	?	?
C_5	0	0	1	1	?	?

Symmetry Breaking: Finite combination of row and column symmetries

- FV, PR, L, LX,
- FVPR, FVL, FVLX, PRL, PRLX, LPR, LLX,
- FVPRL, FVPRLX, FVLPR, FVLLX, PRLLX, LPRLX,
- FVPRLLX, FVLPRLX

Example (PRLX (Wordpress with 3 Wordpress instances))

The assignment matrix:

	V_1	V_2	V_3	V_4	V_5	V_6
C_1	1	1	1	0	0	0
C_2	1	1	0	0	0	0
C_3	0	0	0	0	0	0
C_4	0	0	0	1	0	0
C_5	0	0	0	0	1	1

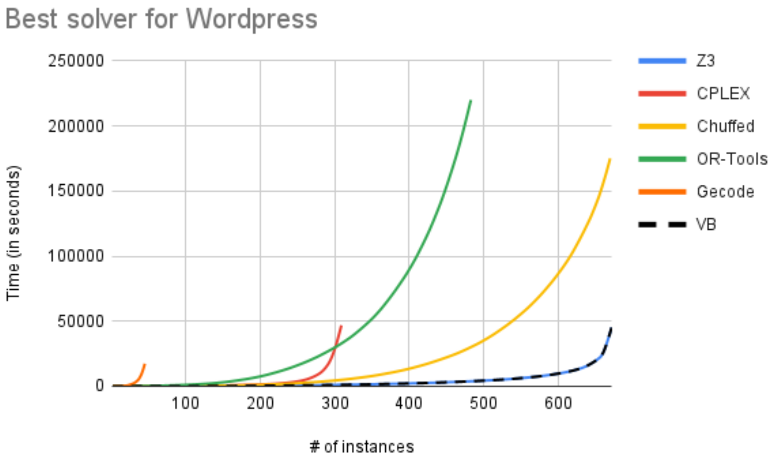
The price vector: $p = [379, 379, 210, 210, 210, 210]$.

Symmetry breakers:

$$P_1 \geq P_2 \wedge$$
$$P_1 = P_2 \Rightarrow a_{11} \geq a_{12} \wedge$$
$$P_1 = P_2 \wedge a_{11} \geq a_{12} \Rightarrow a_{21} \geq a_{22} \wedge$$
$$P_1 = P_2 \wedge a_{11} = a_{12} \Rightarrow a_{31} = a_{32} \wedge$$
$$P_2 \geq P_3 \ \wedge \ldots$$

Contents

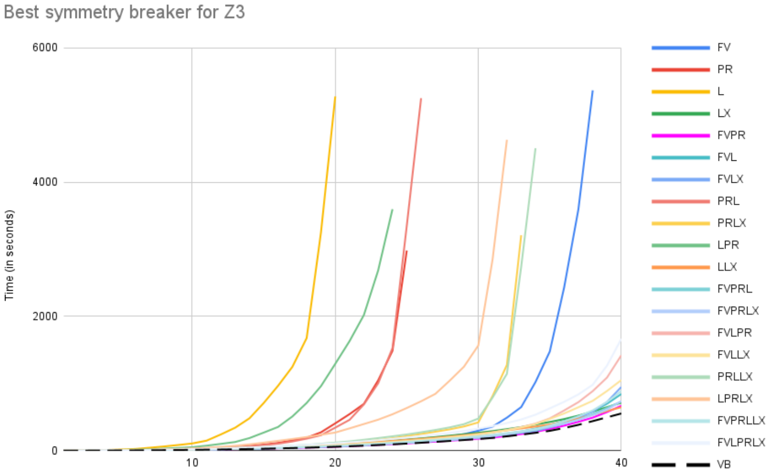
Experimental Results II



Symmetry Breaking: Row-Column Symmetries (cont'd)

Best symmetry breaker for Z3: FVPR

Remark: Combination of more than two symmetry breakers did not lead to better results although more symmetries are broken. This means that breaking more symmetries does not necessarily mean a computational improvement, since more more constraints are added.



Contents

Conclusions

- We proposed several strategies to tackle the scalability issues in the case of optimal deployment of Wordpress application in the Cloud.

Conclusions

- We proposed several strategies to tackle the scalability issues in the case of optimal deployment of Wordpress application in the Cloud.
- The Wordpress problem:

Conclusions

- We proposed several strategies to tackle the scalability issues in the case of optimal deployment of Wordpress application in the Cloud.
- The Wordpress problem:
  - was accepted at the CSPLib initiative `https://www.csplib.org/Problems/prob090/`

Conclusions

- We proposed several strategies to tackle the scalability issues in the case of optimal deployment of Wordpress application in the Cloud.
- The Wordpress problem:
  - was accepted at the CSPLib initiative `https://www.csplib.org/Problems/prob090/`
  - was submitted to the Minizinc 2022 Challenge `https://www.minizinc.org/challenge2022/challenge.html`

Conclusions

- We proposed several strategies to tackle the scalability issues in the case of optimal deployment of Wordpress application in the Cloud.
- The Wordpress problem:
  - was accepted at the CSPLib initiative `https://www.csplib.org/Problems/prob090/`
  - was submitted to the Minizinc 2022 Challenge `https://www.minizinc.org/challenge2022/challenge.html`
  - available on Github `https://github.com/BogdanD02/Cloud-Resource-Provisioning` release V1.0.0 together with experimental results for other case studies

Conclusions

- We proposed several strategies to tackle the scalability issues in the case of optimal deployment of Wordpress application in the Cloud.
- The Wordpress problem:
  - was accepted at the CSPLib initiative `https://www.csplib.org/Problems/prob090/`
  - was submitted to the Minizinc 2022 Challenge `https://www.minizinc.org/challenge2022/challenge.html`
  - available on Github `https://github.com/BogdanD02/Cloud-Resource-Provisioning` release V1.0.0 together with experimental results for other case studies
- Using symmetry breakers, we were able to solve use cases with up to 30 component instances and 500 Cloud offers.

Conclusions

- We proposed several strategies to tackle the scalability issues in the case of optimal deployment of Wordpress application in the Cloud.
- The Wordpress problem:
  - was accepted at the CSPLib initiative `https://www.csplib.org/Problems/prob090/`
  - was submitted to the Minizinc 2022 Challenge `https://www.minizinc.org/challenge2022/challenge.html`
  - available on Github `https://github.com/BogdanD02/Cloud-Resource-Provisioning` release V1.0.0 together with experimental results for other case studies
- Using symmetry breakers, we were able to solve use cases with up to 30 component instances and 500 Cloud offers.
- *Observation*: Number of variables and clauses is not really relevant as they can actually easy the problem to be solved.

Discussion

Discussion

- Application of the solution in practice (with Vlad Luca)

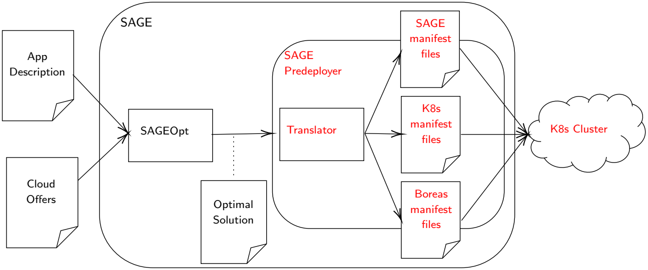


Figure: SAGE General Architecture

Discussion

- Application of the solution in practice (with Vlad Luca)

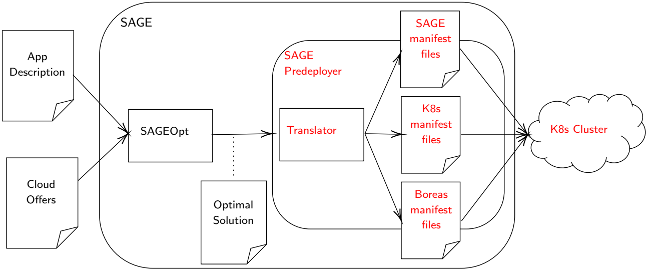


Figure: SAGE General Architecture

- Use graph neural networks for handling the third step of the *automated deployment* (dynamic modification to cope with peaks of user requests) (with Eduard Laitin)

Discussion

- Application of the solution in practice (with Vlad Luca)

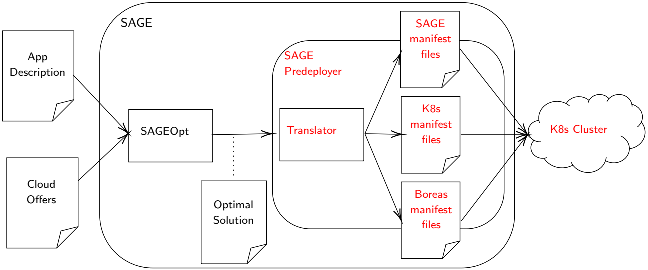


Figure: SAGE General Architecture

- Use graph neural networks for handling the third step of the *automated deployment* (dynamic modification to cope with peaks of user requests) (with Eduard Laitin)
- A more formal explanation why combination of more symmetry breakers does not improve the computational time

Discussion

- Application of the solution in practice (with Vlad Luca)

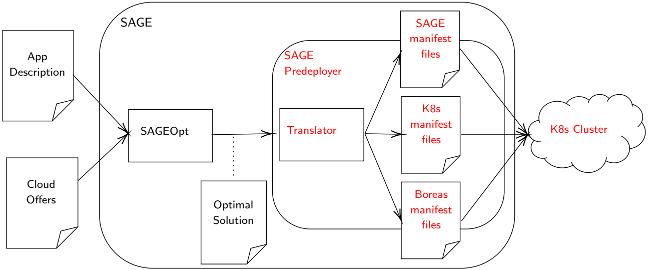


Figure: SAGE General Architecture

- Use graph neural networks for handling the third step of the *automated deployment* (dynamic modification to cope with peaks of user requests) (with Eduard Laitin)
- A more formal explanation why combination of more symmetry breakers does not improve the computational time
  - **Barrier:** static symmetry breakers interfere with the black-box implementation of an SMT solver.

Discussion

- Application of the solution in practice (with Vlad Luca)

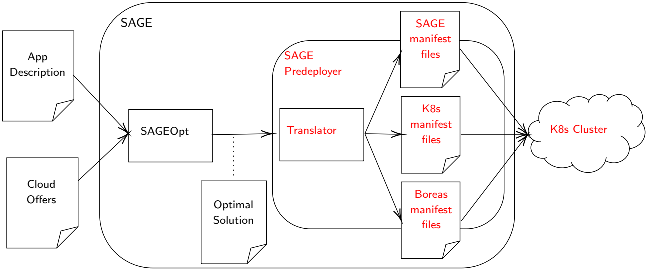


Figure: SAGE General Architecture

- Use graph neural networks for handling the third step of the *automated deployment* (dynamic modification to cope with peaks of user requests) (with Eduard Laitin)
- A more formal explanation why combination of more symmetry breakers does not improve the computational time
  - **Barrier:** static symmetry breakers interfere with the black-box implementation of an SMT solver.
  - **Solution:** Dynamic symmetry breakers.