Symmetry Breaking for the Cloud Resource Allocation Problem

Francesca Drăguț, Adelina Anescu

Faculty of Mathematics and Informatics
West University of Timișoara

January 30, 2022

Introduction.

Cloud Resource Allocation problem: deploying component-based applications on virtual machines.

Constraint Satisfaction Problem (CSP): we are checking if there is any satisfiable assignment of components on virtual machines.

Constraint Optimization Problem (COP): we are minimizing the total price for the virtual machines.

Our aim: describe and develop symmetry breaking methods for the Cloud Resource Allocation problem. Also, identify the most feasible technique.

Introduction.

Cloud Resource Allocation problem: deploying component-based applications on virtual machines.

Constraint Satisfaction Problem (CSP): we are checking if there is any satisfiable assignment of components on virtual machines.

Constraint Optimization Problem (COP): we are minimizing the total price for the virtual machines.

Our aim: describe and develop symmetry breaking methods for the Cloud Resource Allocation problem. Also, identify the most feasible technique.

Introduction.

Cloud Resource Allocation problem: deploying component-based applications on virtual machines.

Constraint Satisfaction Problem (CSP): we are checking if there is any satisfiable assignment of components on virtual machines.

Constraint Optimization Problem (COP): we are minimizing the total price for the virtual machines.

Our aim: describe and develop symmetry breaking methods for the Cloud Resource Allocation problem. Also, identify the most feasible technique.

Introduction.

Cloud Resource Allocation problem: deploying component-based applications on virtual machines.

Constraint Satisfaction Problem (CSP): we are checking if there is any satisfiable assignment of components on virtual machines.

Constraint Optimization Problem (COP): we are minimizing the total price for the virtual machines.

Our aim: describe and develop symmetry breaking methods for the Cloud Resource Allocation problem. Also, identify the most feasible technique.

Theoretical Analysis. Define Symmetries.

The formalization of a Cloud Resource Allocation problem is an assignment matrix:

	VM1	VM2	VM3	VM4
C1	1	0	0	0
C2	0	0	1	0
C3	0	0	0	1

Table: Example of an assignment matrix for an application with 3 components, that need to be deployed on 4 virtual machines. We assign 1 if the component is on the VM and 0, otherwise.

Problem: we obtain a set of satisfiable matrices, out of which some may be symmetric → large search space, while looking for the one with the lowest cost.

Solution: add symmetry breakers to diminish search space.

Theoretical Analysis. Define Symmetries.

The formalization of a Cloud Resource Allocation problem is an assignment matrix:

	VM1	VM2	VM3	VM4
C1	1	0	0	0
C2	0	0	1	0
C3	0	0	0	1

Table: Example of an assignment matrix for an application with 3 components, that need to be deployed on 4 virtual machines. We assign 1 if the component is on the VM and 0, otherwise.

Problem: we obtain a set of satisfiable matrices, out of which some may be symmetric → large search space, while looking for the one with the lowest cost.

Solution: add symmetry breakers to diminish search space.

Theoretical Analysis. Define Symmetries.

The formalization of a Cloud Resource Allocation problem is an assignment matrix:

	VM1	VM2	VM3	VM4
C1	1	0	0	0
C2	0	0	1	0
C3	0	0	0	1

Table: Example of an assignment matrix for an application with 3 components, that need to be deployed on 4 virtual machines. We assign 1 if the component is on the VM and 0, otherwise.

Problem: we obtain a set of satisfiable matrices, out of which some may be symmetric → large search space, while looking for the one with the lowest cost.

Solution: add symmetry breakers to diminish search space.

Theoretical Analysis. Define Symmetries.

The formalization of a Cloud Resource Allocation problem is an assignment matrix:

	VM1	VM2	VM3	VM4
C1	1	0	0	0
C2	0	0	1	0
C3	0	0	0	1

Table: Example of an assignment matrix for an application with 3 components, that need to be deployed on 4 virtual machines. We assign 1 if the component is on the VM and 0, otherwise.

Problem: we obtain a set of satisfiable matrices, out of which some may be symmetric → large search space, while looking for the one with the lowest cost.

Solution: add symmetry breakers to diminish search space.

Theoretical Analysis. Symmetry breaking techniques.

Three types: reformulation, static symmetry breaking, dynamic symmetry breaking [1].

Our choice: static symmetry breaking - operates with symmetry breaking constraints.

Static symmetry breaking constraints: we worked with:

- *Lexicographic ordering*: each 2 adjacent columns are in decreasing order.
- *Fixed Values*: fix values before search, considering component conflicts.

Theoretical Analysis. Symmetry breaking techniques.

Three types: reformulation, static symmetry breaking, dynamic symmetry breaking [1].

Our choice: static symmetry breaking - operates with symmetry breaking constraints.

Static symmetry breaking constraints: we worked with:

- *Lexicographic ordering*: each 2 adjacent columns are in decreasing order.
- *Fixed Values*: fix values before search, considering component-conflicts.

Theoretical Analysis. Symmetry breaking techniques.

Three types: reformulation, static symmetry breaking, dynamic symmetry breaking [1].

Our choice: static symmetry breaking - operates with symmetry breaking constraints.

Static symmetry breaking constraints: we worked with:

- *Lexicographic ordering*: each 2 adjacent columns are in decreasing order.
- *Fixed Values*: fix values before search, considering component-conflicts.
- *Price-based ordering*: sort virtual machines by price.

Theoretical Analysis. Symmetry breaking techniques.

Three types: reformulation, static symmetry breaking, dynamic symmetry breaking [1].

Our choice: static symmetry breaking - operates with symmetry breaking constraints.

Static symmetry breaking constraints: we worked with:

- *Lexicographic ordering*: each 2 adjacent columns are in decreasing order.
- *Fixed Values*: fix values before search, considering component-conflicts.
- *Price-based ordering*: sort virtual machines by price.

Theoretical Analysis. Symmetry breaking techniques.

Three types: reformulation, static symmetry breaking, dynamic symmetry breaking [1].

Our choice: static symmetry breaking - operates with symmetry breaking constraints.

Static symmetry breaking constraints: we worked with:

- *Lexicographic ordering*: each 2 adjacent columns are in decreasing order.
- *Fixed Values*: fix values before search, considering component-conflicts.
- *Price-based ordering*: sort virtual machines by price.

Theoretical Analysis. Symmetry breaking techniques.

Three types: reformulation, static symmetry breaking, dynamic symmetry breaking [1].
Our choice: static symmetry breaking - operates with symmetry breaking constraints.

Static symmetry breaking constraints: we worked with:

- *Lexicographic ordering*: each 2 adjacent columns are in decreasing order.
- *Fixed Values*: fix values before search, considering component-conflicts.
- *Price-based ordering*: sort virtual machines by price.

Theoretical Analysis. Secure Web Container Example.

Secure Web Container: real-life example of component-based application, which needs to be deployed in Cloud.

Given: 5 components, 6 virtual machines, each with 20 possible types ⇒ 120 offers.

Application-specific constraints:

- C1 and C4 in conflict with all other components, C2 in conflict with C3;
- C1 on exactly one VM;
- C2 and C3 together must appear at least 3 times;
- C5 on each machine, except where C1 or C4 already placed;
- Each 10 deployments of C5 ⇒ 1 deployment of C4.

Theoretical Analysis. Secure Web Container Example.

Secure Web Container: real-life example of component-based application, which needs to be deployed in Cloud.
Given: 5 components, 6 virtual machines, each with 20 possible types ⇒ 120 offers.

Application-specific constraints:

- C1 and C4 in conflict with all other components, C2 in conflict with C3;
- C1 on exactly one VM;
- C2 and C3 together must appear at least 3 times;
- C5 on each machine, except where C1 or C4 already placed;
- Each 10 deployments of C5 ⇒ 1 deployment of C4.

Theoretical Analysis. Secure Web Container Example.

Secure Web Container: real-life example of component-based application, which needs to be deployed in Cloud.
Given: 5 components, 6 virtual machines, each with 20 possible types ⇒ 120 offers.

Application-specific constraints:

- C1 and C4 in conflict with all other components, C2 in conflict with C3;
- C1 on exactly one VM;
- C2 and C3 together must appear at least 3 times;
- C5 on each machine, except where C1 or C4 already placed;
- Each 10 deployments of C5 ⇒ 1 deployment of C4.

Experimental Design. The Z3 Environment.

We implemented the encoding of Secure Web Container in the Z3 environment [2].

Z3 = Satisfiability Modulo Theory (SMT) solver → used when working with quantified formulas.

We used the online version of the Z3 software, as it had enough capabilities to support our tests.

We formalized the Z3 encoding and transformed it into mathematical formulas (see next slides).

Experimental Design. The Z3 Environment.

We implemented the encoding of Secure Web Container in the Z3 environment [2].

Z3 = Satisfiability Modulo Theory (SMT) solver → used when working with quantified formulas.

We used the online version of the Z3 software, as it had enough capabilities to support our tests.

We formalized the Z3 encoding and transformed it into mathematical formulas (see next slides).

Experimental Design. The Z3 Environment.

We implemented the encoding of Secure Web Container in the Z3 environment [2].

Z3 = Satisfiability Modulo Theory (SMT) solver → used when working with quantified formulas.

We used the online version of the Z3 software, as it had enough capabilities to support our tests.

We formalized the Z3 encoding and transformed it into mathematical formulas (see next slides).

Experimental Design. The Z3 Environment.

We implemented the encoding of Secure Web Container in the Z3 environment [2].

Z3 = Satisfiability Modulo Theory (SMT) solver → used when working with quantified formulas.

We used the online version of the Z3 software, as it had enough capabilities to support our tests.

We formalized the Z3 encoding and transformed it into mathematical formulas (see next slides).

Experimental Design. Adding Symmetry Breakers.

LEXICOGRAPHIC ORDERING

We used column-wise lexicographic ordering → orders columns decreasingly.
By ordering the matrices, the symmetric ones will become undistinguishable.

	VM1	VM2	VM3
C1	0	1	0
C2	0	1	1
C3	0	1	1
C4	1	0	0

→

	VM1	VM2	VM3
C1	1	0	0
C2	1	1	0
C3	1	1	0
C4	0	0	1

Table: How the assignment matrix changes after applying LX column-wise.

Experimental Design. Adding Symmetry Breakers.

FIXED VALUES

This method requires some prior deductive reasoning.
Fixing the values depends on component-conflicts.

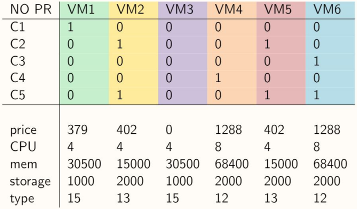
	VM1	VM2	VM3	VM4	VM5	VM6
C1	1	0	0	0	0	0
C2	0	0	1	0		
C3	0	0	0	1		
C4	0	1	0	0		
C5	0	0				

Table: The fixed values in the matrix, for the Secure Web Container example, before the search starts.

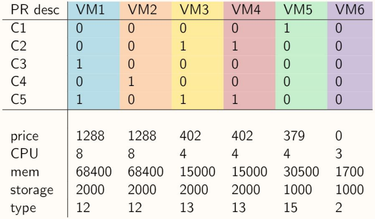
Experimental Design. Adding Symmetry Breakers.

PRICE-BASED ORDERING

For the price-based ordering technique, the prices of the virtual machines are set to be in some kind of order.
When prices are ordered, the search time reduces → search will be linear.



NO PR	VM1	VM2	VM3	VM4	VM5	VM6
C1	1	0	0	0	0	0
C2	0	1	0	0	1	0
C3	0	0	0	0	0	1
C4	0	0	0	1	0	0
C5	0	1	0	0	1	1
price	379	402	0	1288	402	1288
CPU	4	4	4	8	4	8
mem	30500	15000	30500	68400	15000	68400
storage	1000	2000	1000	2000	2000	2000
type	15	13	15	12	13	12



PR desc	VM1	VM2	VM3	VM4	VM5	VM6
C1	0	0	0	0	1	0
C2	0	0	1	1	0	0
C3	1	0	0	0	0	0
C4	0	1	0	0	0	0
C5	1	0	1	1	0	0
price	1288	1288	402	402	379	0
CPU	8	8	4	4	4	3
mem	68400	68400	15000	15000	30500	1700
storage	2000	2000	2000	2000	1000	1000
type	12	12	13	13	15	2

Results.

Comparison of the execution times for the Secure Web Container example, without symmetry breaking techniques, with price-based ordering (PR), lexicographical ordering (LX), fixed values (FV) and their variants.

Symm. Breakers	Time (s)
No Symm. Br.	0.779
PR (desc.)	0.390
PR (asc.)	0.424
LX	0.470
FV (all constraints)	0.352
FV (elim. constr.)	0.458

Conclusions and Future Work.

We studied the Cloud Resource Allocation Problem, with the main focus being on symmetry breaking techniques → Secure Web Container exmaple.

Our aim: show how symmetry breakers improve execution time and which one is the most feasible technique.

Did we reach it?: yes, we proved that symmetry breakers diminish the execution time for the Secure Web Container; we learned that the most feasible technique for this example is PR in descending order.

Conclusions and Future Work.

We studied the Cloud Resource Allocation Problem, with the main focus being on symmetry breaking techniques → Secure Web Container exmaple.

Our aim: show how symmetry breakers improve execution time and which one is the most feasible technique.

Did we reach it?: yes, we proved that symmetry breakers diminish the execution time for the Secure Web Container; we learned that the most feasible technique for this example is PR in descending order.

Conclusions and Future Work.

We studied the Cloud Resource Allocation Problem, with the main focus being on symmetry breaking techniques → Secure Web Container exmaple.

Our aim: show how symmetry breakers improve execution time and which one is the most feasible technique.

Did we reach it?: yes, we proved that symmetry breakers diminish the execution time for the Secure Web Container; we learned that the most feasible technique for this example is PR in descending order.

Conclusions and Future Work.

What needs to be improved?: find a more precise and reliable algorithm for row-wise lexicographic ordering.

Future Work: add more symmetry breakers in the study, in order to see how they behave and to see if we can find an even better one.

Conclusions and Future Work.

What needs to be improved?: find a more precise and reliable algorithm for row-wise lexicographic ordering.

Future Work: add more symmetry breakers in the study, in order to see how they behave and to see if we can find an even better one.

Demo

We are using the encodings for Secure Web Container from [3] and [4].

References

[1] Erascu, Madalina and Micota, Flavia and Zaharie, Daniela (2021)
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, volume 121.

[2] Z3 Online Demonstrator
`https://compsys-tools.ens-lyon.fr/z3/`

[3] GitHub Repository for MANeUveR Project
`https://github.com/Maneuver-PED`

[4] GitHub Repository for our project
`https://github.com/francescadragut/Symmetry-Breaking-for-the-Cloud-Resource-Allocation-Problem`

Thank You!