A Scalable Hybrid Approach for Applications Placement in the Cloud

Mădălina Erașcu[1] Flavia Micota[2] Daniela Zaharie[2]

[1]Institute e-Austria Timișoara, Romania

[2]West University of Timișoara, Romania

zflavia@info.uvt.ro

October 28, 2015

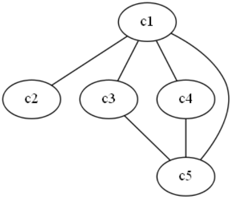
This research was partially supported by the Romanian national grant PN-II-ID-PCE-2011-3-0260 (AMICAS) and by the grant of the European Commission FP7 project SPECS (Grant Agreement no. 610795).

Content

Motivation

Consider the architecture of a web application ensuring a secure-billing e-mail service composed of the following components:

- a coding service (c_1)
- a software manager of the user rights and privileges (c_2)
- a gateway component (c_3)
- a SQL server (c_4)
- a load balancer (c_5)



We want to

- Deploy each component on at least one virtual machine (VM)
- Ensure that there are no conflicting components placed on the same VM
- Minimize the number of VMs hosting at least one component

We need to solve a *resource allocation problem in the Cloud!*

Problem

Problem Description

Input:a set C of N components: $C = \{c_1, ..., c_N\}$ and a set of M VMs $M = \{v_1, ..., v_M\}$
Objective: *minimize* the number of VMs required to deploy components such that:

C1: There are no conflicting components placed on the same VM

C2: Each component is placed on at least one VM

C3: Several instances of a component can be deployed (to ensure a given redundancy level)

C4: The number of VMs having at least one component is minimal

What type of problem is?

- Constrained optimization problem

How to solve it?

- What type of approach to choose to solve the problem?
  - Exact solution
  - Approximate solution
- What representation to choose for the problem?

Problem

Problem Representation

Input:a set C of N components: $C = \{c_1, ..., c_N\}$ and a set of M VMs $V = \{v_1, ..., v_M\}$
Objective: *minimize* the number of VMs required to deploy the N components such that:

C1: There are no conflicting components placed on the same VM

Restriction matrix between components, $R_{N \times N}$

$$R_{ij} = \begin{cases} 0, & c_i \text{ not in conflict with } c_j \\ 1, & c_i \text{ in conflict with } c_j \text{ or } i = j \end{cases}$$

C2: Each component is placed on at least one VM

The number of VM is greater or equal with the number of components $M \geq N$

C3: A component can be deployed on several VMs

Number of instances of component c_i equals n_i, $n_i \geq 1$

Representation

Integer Quadratic Problem

Aim: Find a set of binary variables a_{ik} with $i = \overline{1, N}$, $k = \overline{1, M}$

$$a_{ik} = \begin{cases} 1, & \text{component } c_i \text{ is allocated to VM } v_k \\ 0, & \text{component } c_i \text{ is not allocated to VM } v_k \end{cases}$$

C1: There are no conflicting components placed on the same VM

$$\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ik} a_{jk} R_{ij} = 1, \ k = \overline{1, M}$$

C2: Each component is placed on at least to one VM

C3: Several instances of a component can be deployed

$$\sum_{k=1}^{M} a_{ik} = n_i, \ n_i \geq 1, \ i = \overline{1, N}$$

C4: The number of VMs having at least one component is minimal

$$\sum_{k=1}^{M} max_{i=\overline{1,N}} a_{ik} \text{ should be minimized}$$

Representation

Integer Linear Problem

Aim: Find a set of binary variables a_{ik} with $i = \overline{1, N}$, $k = \overline{1, M}$ and b_k with $k = \overline{1, M}$

$$a_{ik} = \begin{cases} 1, & \text{component } c_i \text{ is allocated to VM } v_k \\ 0, & \text{component } c_i \text{ is not allocated to VM } v_k \end{cases} \qquad b_k = \begin{cases} 1, & \text{VM } v_k \text{ is used} \\ 0, & \text{VM } v_k \text{ is not used} \end{cases}$$

C1: There are no conflicting components placed on the same VM

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

$$a_{ik} \leq b_k, i = \overline{1, N}, k = \overline{1, M}$$

C2: Each component is allocated on at least to one VM

C3: A component can be deployed on several VMs

$$\sum_{k=1}^{M} a_{ik} = n_i, \; n_i \geq 1, \; i = \overline{1, N}$$

C4: The number of VMs having at least one component is minimal

$$\sum_{k=1}^{M} b_k \text{ should be minimized}$$

Comparative Analysis

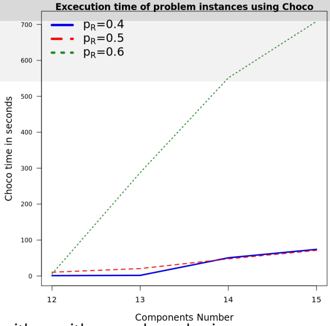
Integer Quadratic Problem

- Search Space
  - Matrix $A_{N \times M}$ that contains the planning
  - Search space size: 2^{NM}
- Number of constraints
  - C1: components are in/not in conflict $\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ik} a_{jk} R_{ij} = 1, \; k = \overline{1, M}$
    - M constraints
  - C2 & C3: number of components $\sum_{k=1}^{M} a_{ik} = n_i, \; n_i \geq 1, \; i = \overline{1, N}$
    - N constraints
  - Total number of constraints: $M + N$

Integer Linear Problem

- Search Space
  - Matrix $A_{N \times M}$ that contains the allocation and an allocated VMs vector b
  - Search space size: 2^{NM+M}
- Number of constraints
  - C1: components are in/not in conflict $a_{ik} + a_{jk} \leq 1, \forall i < j$ s.t. $R_{ij} = 1, k = \overline{1, M}$; $a_{ik} \leq b_k, i = \overline{1, N}, k = \overline{1, M}$
    - $M \cdot |R| + M \cdot N$ constraints, where $|R|$ denotes the number of pairs (i, j) satisfying $i < j$ and $R_{ij} = 1$
  - C2 & C3: number of components $\sum_{k=1}^{M} a_{ik} = n_i, \; n_i \geq 1, \; i = \overline{1, N}$
    - N constraints
  - Total number of constraints: $M \cdot |R| + M \cdot N + N$

Approaches to Solve the Problem



Exact Approach

The search space is explored in a systematic way.

- Problem Representation
  - Integer Linear Optimization Problem
- Solution
  - Choco Solver [http://choco-solver.org/]
    - Explores the search space by alternating constraint filtering algorithms with a search mechanism
  - Disadvantages
    - For problems with a large number of instances it takes a long time to find a solution

Heuristic Approach

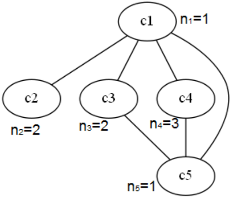
- Problem Representation
  - Integer Quadratic Optimization Problem
- Solution
  - Evolutionary Search
  - Hybrid Search (Evolutionary Search combined with Choco search)

Heuristic Approach

Evolutionary Algorithm

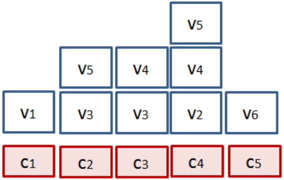
Population Initialization
Repair and evaluate solution
while ⟨Stopping criteria not satisfied⟩ **do**
    Apply crossover
    Apply mutation
    Apply selection
end while

Conflict graph example



Population Elements Encoding

- A solution is represented by N lists each one containing the ID of all VMs on which the corresponding component is placed



Solution Initialization

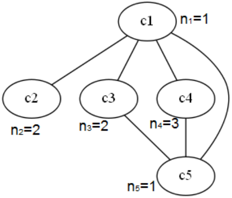
- Random placement such that the constraint C1 (components incompatibility constraint) is not violated

Heuristic Approach

Evolutionary Algorithm

Population Initialization
Repair and evaluate solution
while ⟨Stopping criteria not satisfied⟩ **do**
    Apply crossover
    Apply mutation
    Apply selection
end while

Conflict graph example



Solution repair

- Iteratively move of components
- Until a valid solution is generated or a maximal number of repairing trials is reached

Solution evaluation

- Number of acquired VM
- Number of violated constraints

Stopping criteria

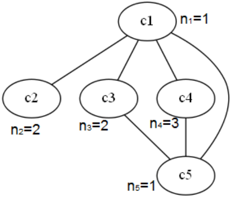
- The best element of the population is not improved after 100 iterations

Heuristic Approach

Evolutionary Algorithm

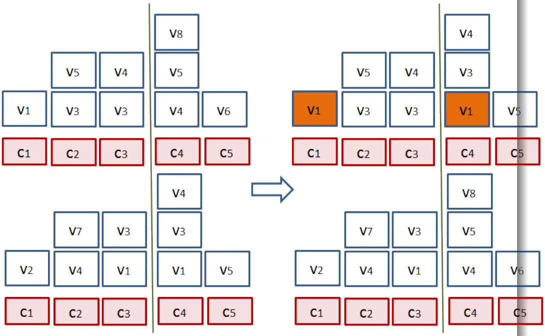
Population Initialization
Repair and evaluate solution
while ⟨Stopping criteria not satisfied⟩ **do**
  Apply crossover
  Apply mutation
  Apply selection
end while

Conflict graph example



Crossover

- One-cut point crossover
  - Apply crossover
  - Repair the resulting element if necessary
  - Evaluate element

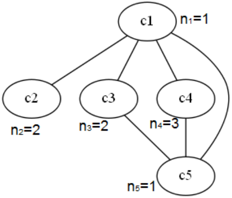


Heuristic Approach

Evolutionary Algorithm

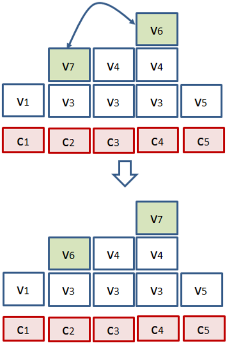
Population Initialization
Repair and evaluate solution
while ⟨Stopping criteria not satisfied⟩ **do**
    Apply crossover
    Apply mutation
    Apply selection
end while

Conflict graph example



Mutation

- Select a component c_i with probability $p_M = 1/N$ and one of a VMs on which it is deployed
  - Remove the component from that VM
  - Move the component to another VM
  - Swap two components placed on different VMs

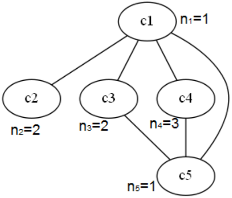


Heuristic Approach

Evolutionary Algorithm

```
Population Initialization
Repair and evaluate solution
while ⟨Stopping criteria not satisfied⟩ do
    Apply crossover
    Apply mutation
    Apply selection
end while
```

Conflict graph example



Selection

- Elitist selection
  - Select from $2 * m$ candidate solutions the best m ones

How are compared two solutions?

- IF both solutions satisfy all constraints THEN the solution involving the smaller number of VMs is considered to be better.
- IF only one of the two solutions satisfies the constraints THEN it is considered better.
- IF both solutions are not feasible THEN the solution that violates the smallest number of constraints is considered better.

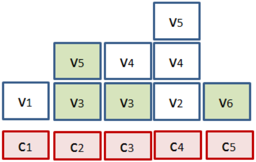
Test Cases

Test case instances generation

- Generated test instances consisting of matrices that encode the conflicts between the components
  - Different degrees of conflicts (controlled by a conflict probability p_R)
  - Multiple component instances (controlled by a probability p_I)
  - The problem becomes harder as p_R and p_I increase

Problem solving variants

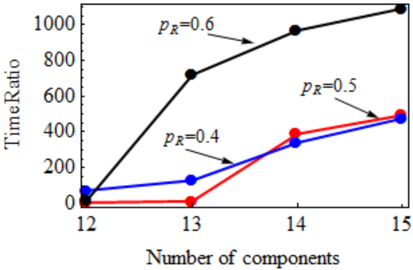
- Exact approach: Choco search
- Heuristic approach: Evolutionary Algorithm (EA)
- Hybrid approach
  - Step 1: Run the EA
  - Step 2: Starting with the solution obtained by the EA a partial systematic search is conducted
    - Reduce the search space size by fixing 50% of the solution components and search only for values corresponding to the other components.



Results

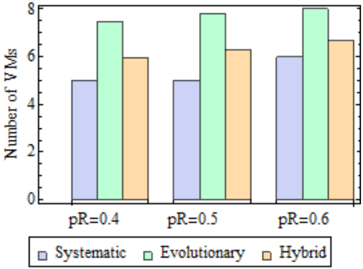
Hybrid vs systematic search

- Significant decrease in time



- Ratio between the running time of systematic search and that of hybrid search

- Slight decrease in quality



- Number of used VMs (average values in the case of evolutionary and hybrid search)

Results

Hybrid vs Systematic Search

Hybrid approach provides feasible solutions significantly faster than the exact solver

- Exact solver has been stopped after 30 minutes with no guarantee of optimality
- The total number of available VMs is $M = 20$

Problem			Systematic search		Hybrid search	
N	p_R	p_I	M_{est}	Time[s]	avg(M)±stdev(M)	avg(Time)[s]
15	0.4	0.1	8	1800	9.17±1.19	2.33
15	0.4	0.5	11	1800	13.33±1.43	2.88
15	0.4	0.9	N/A	1800	15.73±0.91	6.08
15	0.6	1	N/A	1800	17.32±0.60	18.89

Conclusions & Future Work

Conclusions

- Generation of synthetic test cases
- Three approaches to solve the problem:
  - Systematic
  - Evolutionary
  - Hybrid search
- The hybrid approach
  - Provides solutions close to those provided by an exact solver but in a significantly smaller amount of time
  - Being useful especially when the problem complexity precludes the use of systematic search

Future work

- Identify hybrid strategies which exploit the characteristics of the problem to be solved.
- Conduct tests in real case scenarios