Improving Resource Utilization in Cloud using Application Placement Heuristics

Atakan Aral, Tolga Ovatman

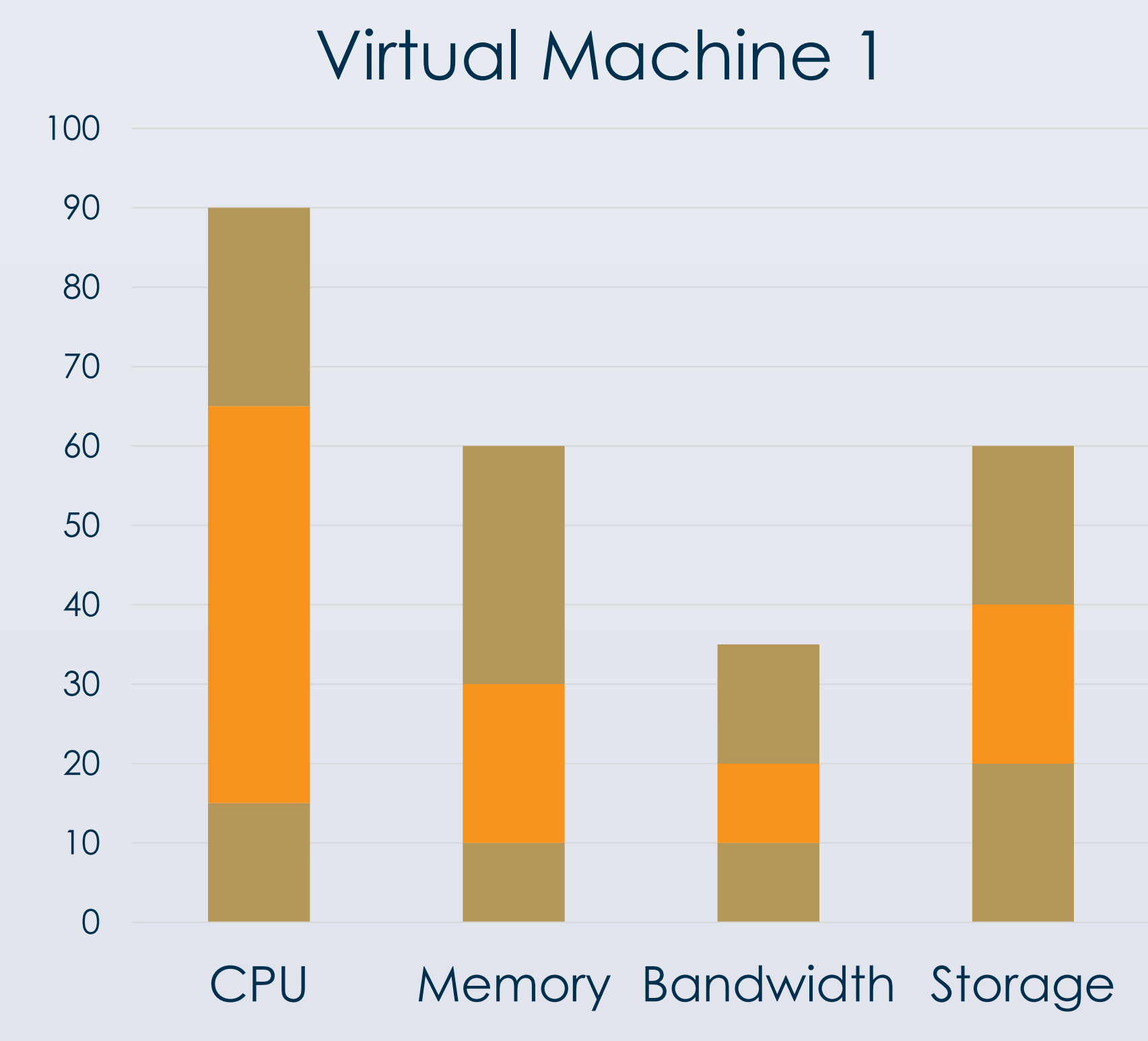
Istanbul Technical University, Department of Computer Engineering

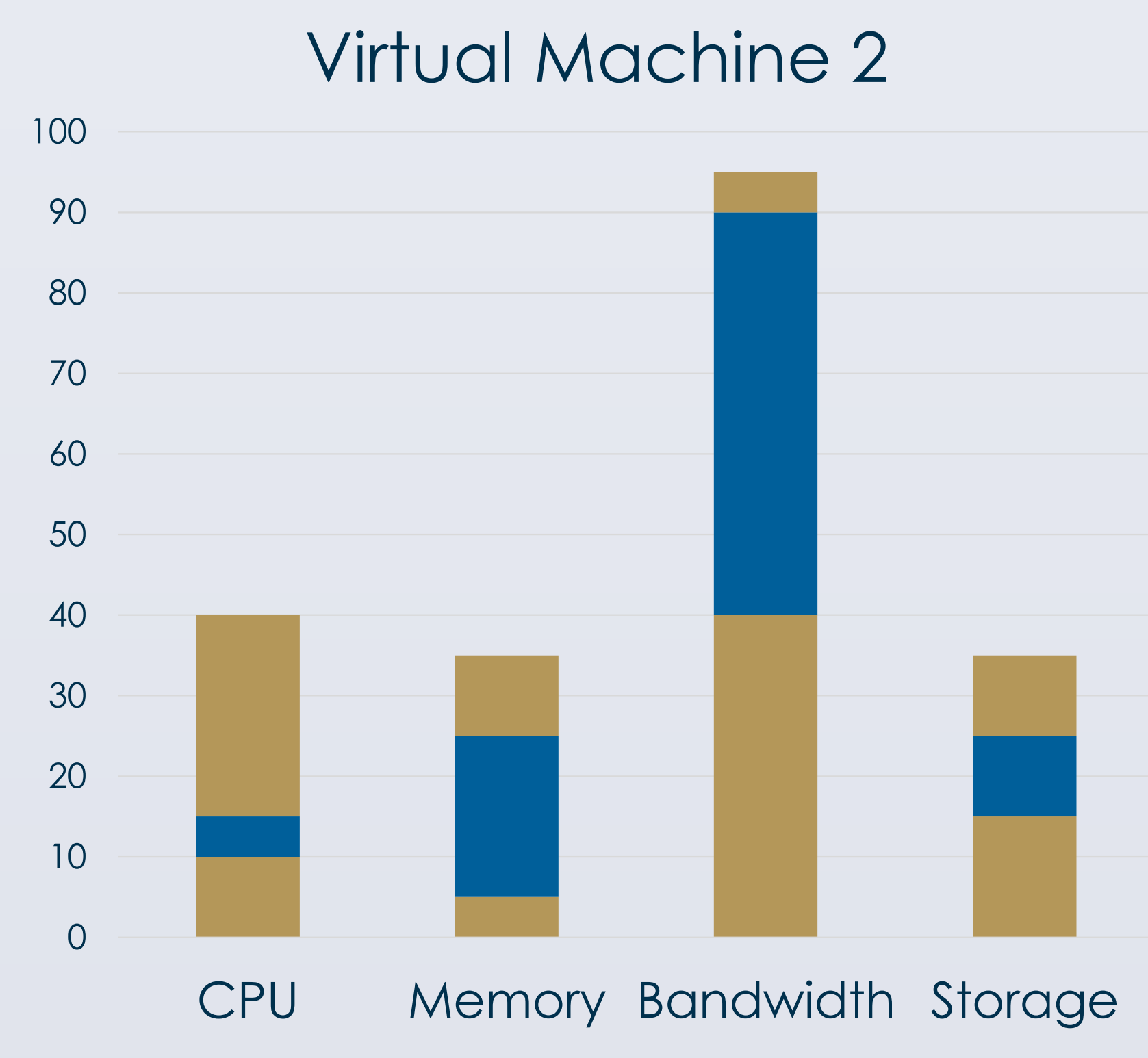
{aralat, ovatman}@itu.edu.tr

ABSTRACT

Application placement is an important concept when providing software as a service in cloud environments. Because of the potential downtime cost of application migration, most of the time additional resource acquisition is preferred over migrating the applications residing in the virtual machines (VMs). This situation results in under-utilized resources. To overcome this problem static/dynamic estimations on the resource requirements of VMs and/or applications can be performed. A simpler strategy is using heuristics during application placement process instead of naively applying greedy strategies like round-robin. In this paper, we propose a number of novel heuristics and compare them with round robin placement strategy and a few proposed placement heuristics in the literature to explore the performance of heuristics in application placement problem. Our focus is to better utilize the resources offered by the cloud environment and at the same time minimize the number of application migrations.

PROBLEM





CANDIDATE SOLUTIONS

- Greedy algorithms (e.g. Round-Robin)
  - Easy to implement
  - Does not produce good placements
- Linear Optimization (e.g. Mixed Integer Programming)
  - Guaranteed to produce optimal placement
  - Computationally expensive
  - Migrations required
- Evenness Heuristics
  - May produce good enough placements
  - There is no universally accepted heuristic for evenness

SUGGESTED SOLUTION

- Assign using an intelligent heuristic when possible
- Do not make any migrations yet
- When no VM is able to hold the arriving application, use Mixed Integer Programming
- Minimize number of migrations as the objective function

```
rejected ← false
while rejected = false do
    receive application A
    assignable ← false
    foreach virtual machine VM do
        if VM has enough capacity for A then
            assignable ← true
            assign A to VM
            calculate unevenness of VM
            remove A from VM
    if assignable = true then
        assign A to VM with min unevenness
    else
        run optimization algorithm
        if optimization succeeds then
            assign A and migrate others
        else rejected ← true
```

CONTRIBUTION

- Novel evenness heuristics
- Comparison of the heuristics with other approaches
- Evaluation of the heuristic approach on various VM counts and capacities
- MIP formulation that minimizes number of migrations

HEURISTICS

- Standard Deviation (*SD*)
  - Natural choice but may miss outliers
- Span (*SP*)
  - Focuses only on the outliers
- Cumulative Difference (*CD*)
  - Similar to SD, only simpler
- CD from Minimum (*DM*)
  - Considers only high outliers
- Skewness (*SK*)
  - Suggested by Xiao et al., 2013

$$SD(v) = \sqrt{\sum_{i=1}^{m} (r_i - \bar{r})^2}$$

$$SP(v) = \max_{i\in[1,m]} (r_i) - \min_{i\in[1,m]} (r_i)$$

$$CD(v) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{m} |r_i - r_j|}{2}$$

$$DM(v) = \sum_{i=1}^{n} \left(r_i - \min_{j\in[1,m]} (r_j) \right)$$

$$SK(v) = \sqrt{\sum_{i=1}^{m} \left(\frac{r_i}{\bar{r}} - 1 \right)^2}$$

v: *A virtual machine,* ***r_i:*** *Utilization of the resource i of v,*
r: *Average utilization of all resources of v,* ***m:*** *Number of resources*

MIP FORMULATION

- **Objective:** Minimize number of migrations

$$\sum_{i=0}^{\#VMs} \sum_{j=0}^{\#Apps} (\text{newAsgn}[i][j] \times \text{oldAsgn}[i][j])$$

- **Constraints:**
  - Each application must be assigned to exactly one VM:

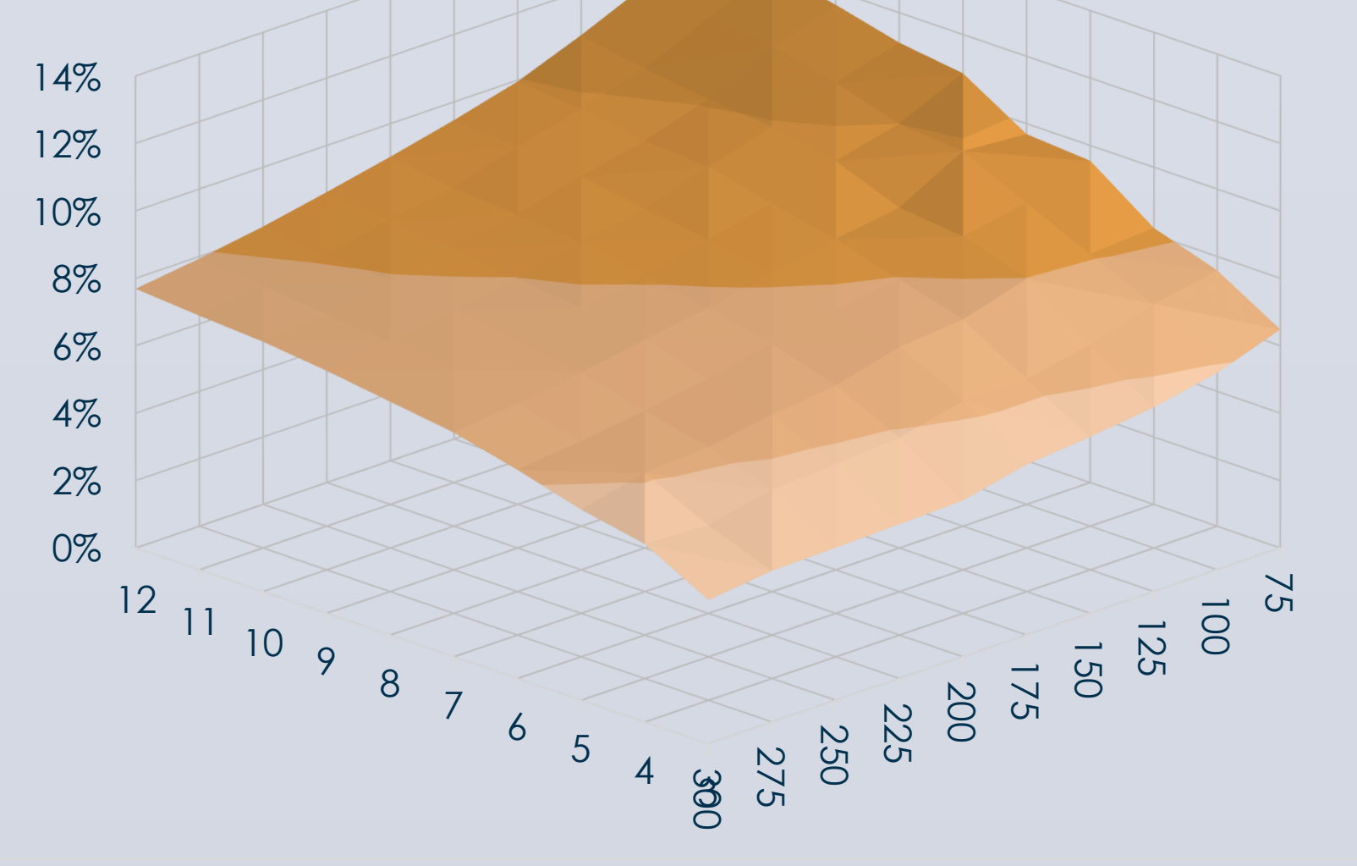
$$\bigwedge_{i=0}^{\#VMs} \left(\sum_{j=0}^{\#Apps} \text{newAsgn}[i][j] = 1 \right)$$

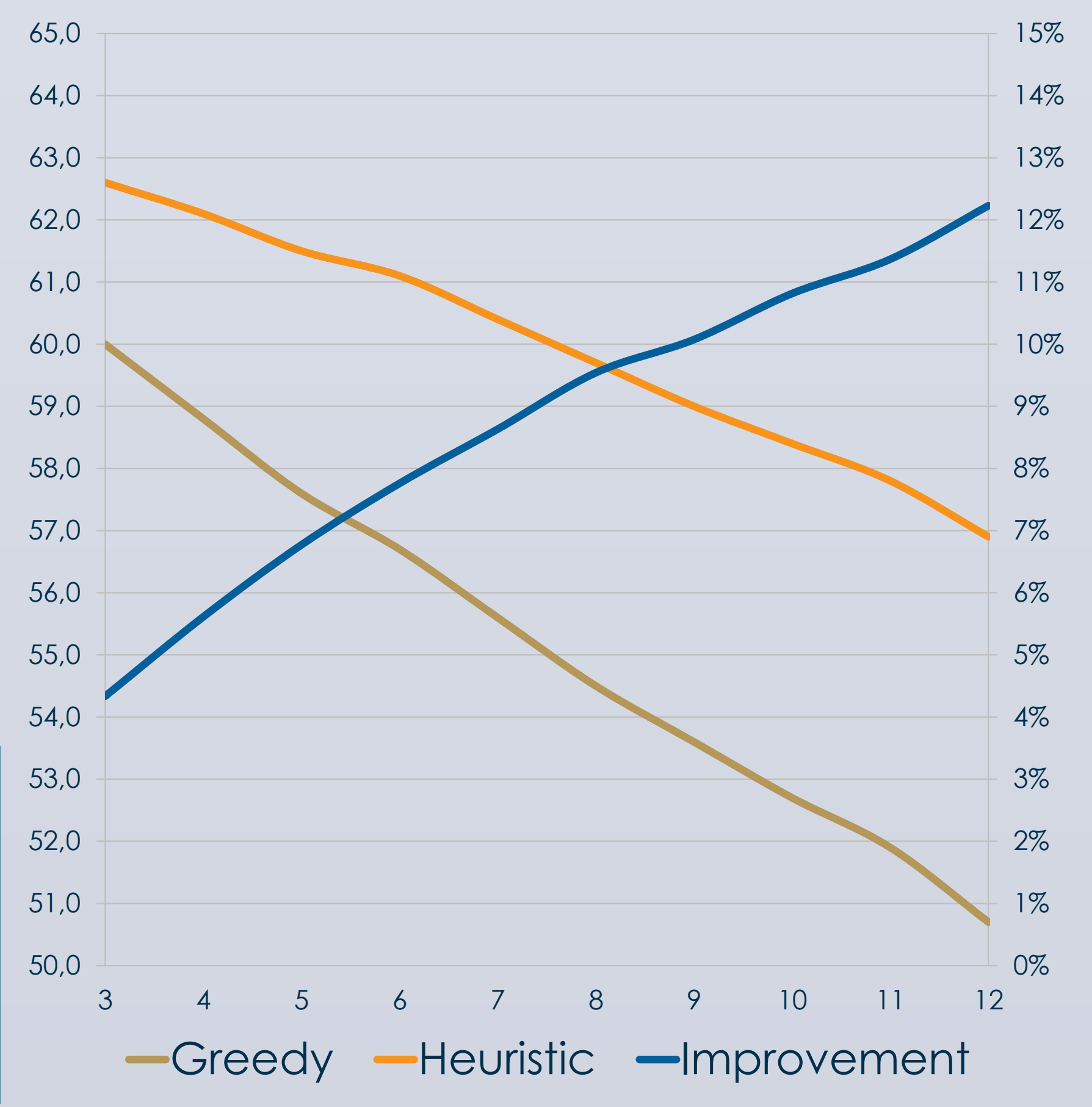
  - No VM should be loaded more than its capacity:

$$\bigwedge_{i=0}^{\#VMs} \bigwedge_{j=0}^{\#Res} \left(\left(\sum_{k=0}^{\#Apps} \text{newAsgn}[i][k] \times \text{resNeed}[j][k] \right) \leq \text{resAv}[i][j] \right)$$

EXPERIMENTAL RESULTS

VM Capacity	**100**	**150**	**200**	**250**	**300**	**200**	**200**	**200**	**200**	**200**
VM Count	**8**	**8**	**8**	**8**	**8**	**4**	**6**	**8**	**10**	**12**
RR	42,0	67,2	92,7	118,4	144,3	46,0	69,3	92,7	116,2	139,7
SK_{min}	45,8	72,3	98,8	125,3	151,9	48,1	73,4	98,8	124,3	149,9
SK_{dec}	45,5	72,0	98,6	125,2	151,9	48,1	73,2	98,6	124,1	149,7
SP_{min}	**46,2**	**73,2**	**100,2**	**127,2**	**154,3**	**48,7**	**74,4**	**100,2**	**126,2**	**152,3**
SP_{dec}	46,0	72,6	99,3	126,0	153,2	48,4	73,8	99,3	124,9	150,7
SD_{min}	**46,2**	**73,2**	**100,2**	**127,3**	**154,3**	**48,7**	**74,4**	**100,2**	**126,2**	**152,3**
SD_{dec}	45,8	72,3	98,9	125,5	152,1	48,2	73,4	98,9	124,4	150,1
CD_{min}	**46,2**	**73,2**	**100,2**	**127,3**	**154,3**	**48,8**	**74,4**	**100,2**	**126,2**	**152,3**
CD_{dec}	45,9	72,5	99,1	125,9	152,7	48,3	73,7	99,1	124,8	150,5
DM_{min}	45,7	72,6	99,6	126,5	153,6	48,3	73,8	99,6	125,4	151,5
DM_{dec}	45,8	72,5	99,2	126,2	153,1	48,3	73,6	99,2	125,0	150,8
Expected	53,3	80,0	106,7	133,3	160,0	53,3	80,0	106,7	133,3	160,0
Improvement	**10,0%**	**8,9%**	**8,1%**	**7,5%**	**6,9%**	**6,1%**	**7,4%**	**8,1%**	**8,6%**	**9,0%**





Strategy	**Avg. Migr. Count**	**Perfect Count**
RR	8,4	26
SD_{min}	5,5	108
SP_{min}	5,6	100
CD_{min}	5,8	86
SK_{min}	7,2	88

- Optimal placement in up to 10,8% of the cases, four times better than greedy
  - No migrations necessary, better service quality
- Delay MIP algorithm up to 12,1% w.r.t. greedy
  - More applications accepted before migrations
- MIP results in 34,5% less application migrations