

Knapsack-based Selection of Cloud Resources with Hierarchical Organizational Dependencies

Authors:

Victor Toporkov

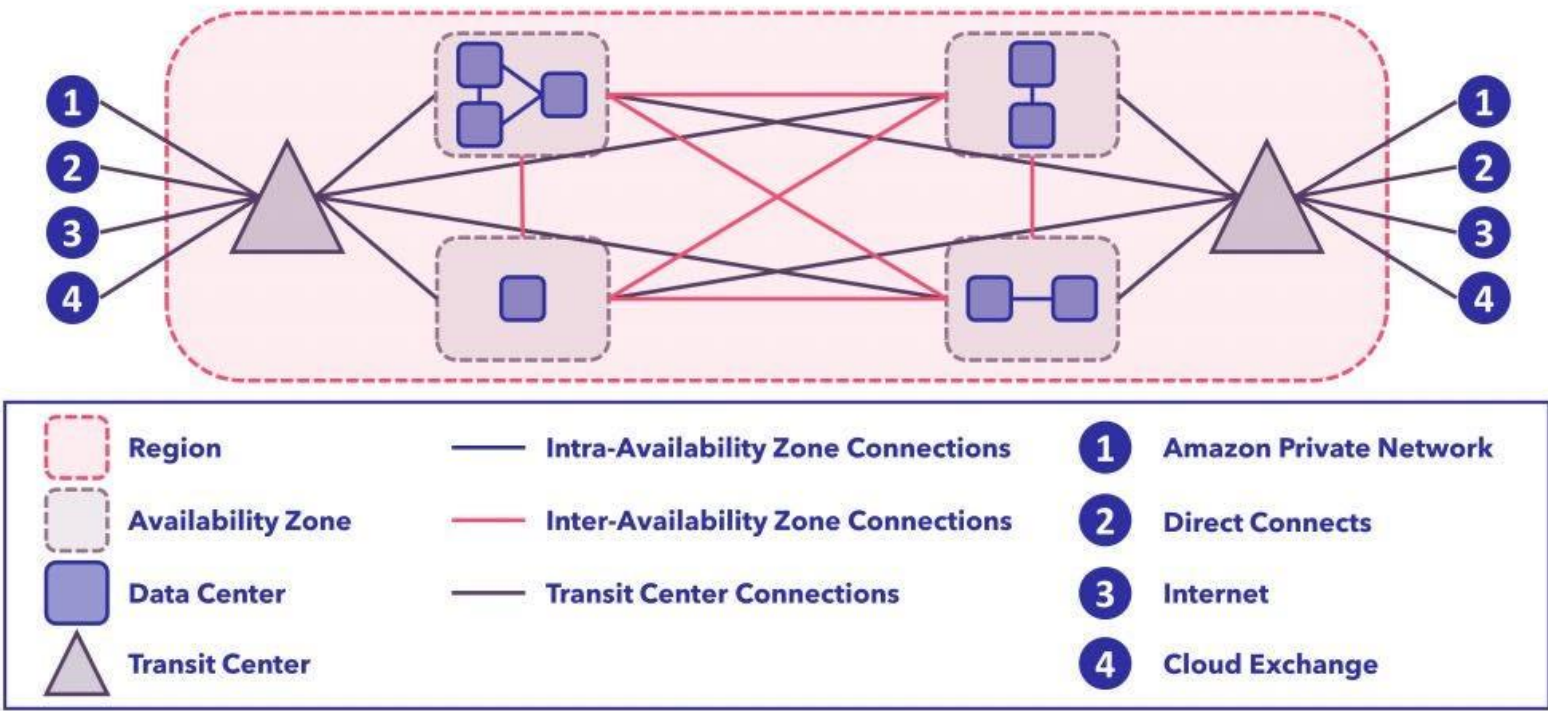
Dmitry Yemelyanov

and Artem Bulkhak

National Research University “MPEI”

- Many modern companies and cloud computing providers offer increasingly sophisticated **multi-tier pricing plans** for their services
 - bonuses, promotions and discounts
 - natural groupings by resource types, network connectivity and geographic distribution
- The resources are physically, logically and naturally clustered in different groups, data centers and geographically separated regions
- Resources selection and scheduling algorithms should account for non-linear pricing and performance models and emerging **dependencies between the available resources**
 - classical optimization algorithms usually operate with homogeneous lists of elements

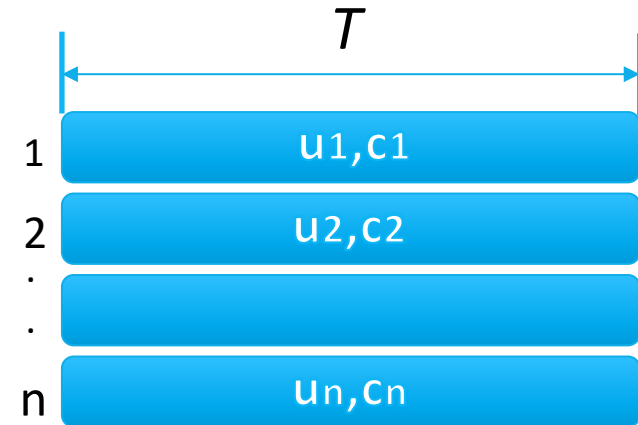
Amazon Web Services - *Region Architecture*



- We model the resource selection problem as a **hierarchical variant of the knapsack problem** to capture group interdependencies, such as tiered pricing structures, incentive bonuses, volume discounts
- We present an **exact dynamic programming-based algorithm** that computes an optimal solution to the problem
 - The algorithm runs in pseudopolynomial time
- We present a simulation study conducted in the **CloudSim environment** to compare solution against brute-force search, branch-and-bound, and heuristic approximation algorithms

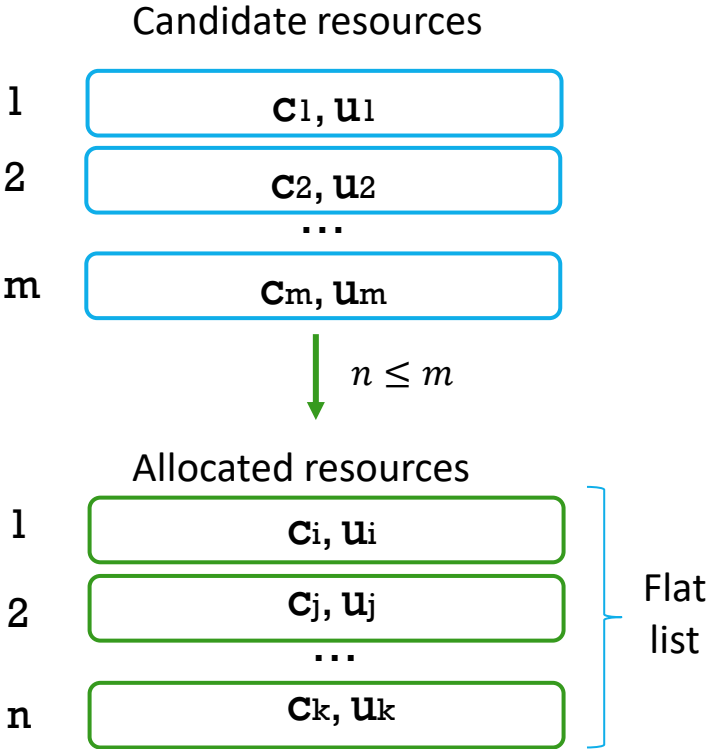
The resource requirements for a single service, parallel job or workflow are arranged into a **resource request**:

- n - number of simultaneously allocated resources
- p - minimal requirements for each resource
- T – total allocation/reservation time period
- C - maximum total job execution cost (budget)



Additionally maximize the total utility criterion

$$U = \sum_{i=1}^n u_i \rightarrow \max$$



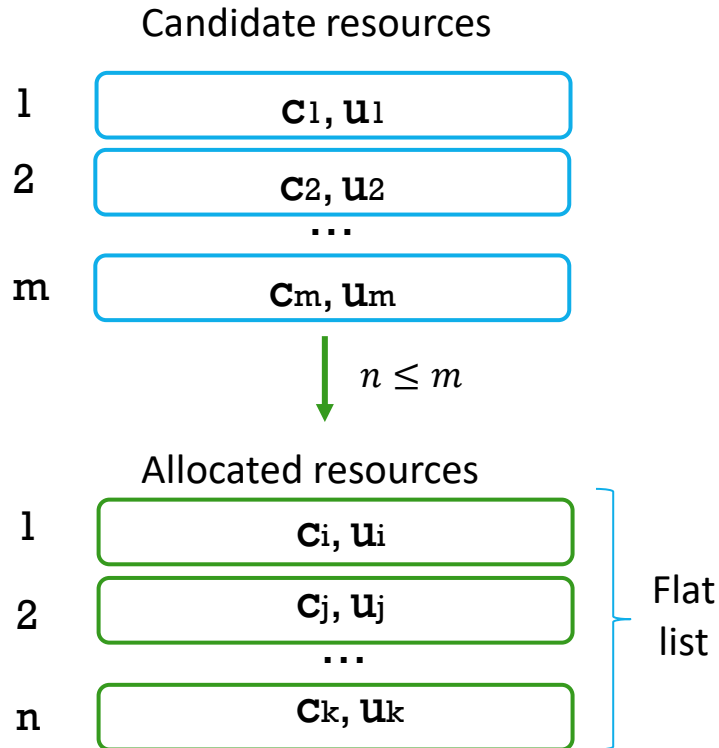
$$U = \sum_{i=1}^m u_i x_i \rightarrow \max$$

$$\sum_{i=1}^m c_i x_i \leq C_j,$$

$$x_i \in \{0,1\}, i = 1..m$$

Number n of allocated resources is not limited: $n \in [0; m]$

Classic 0-1 knapsack problem



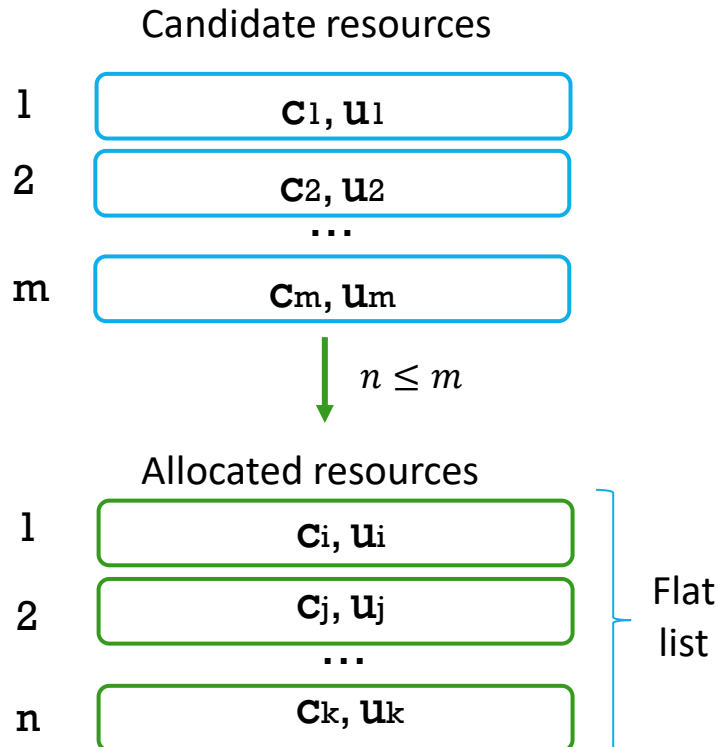
$$U = \sum_{i=1}^m u_i x_i \rightarrow \max$$

$$\sum_{i=1}^m c_i x_i \leq C_j,$$

$$\sum_{i=1}^m x_i = n,$$

$$x_i \in \{0,1\}, i = 1..m$$

Number n of simultaneously required resources is predetermined



$$U = \sum_{i=1}^m u_i x_i \rightarrow \max$$

$$\sum_{i=1}^m c_i x_i \leq C,$$

$$\sum_{i=1}^m x_i \geq n_{\min},$$

$$\sum_{i=1}^m x_i \leq n_{\max},$$

$$x_i \in \{0,1\}, i = 1..m$$

Interval of permissible values $[n_{\min}; n_{\max}]$ is defined for n

This problem describes a more generic resources allocation scenario

$$f_i(c, k) = \max\{f_{i-1}(c, k), f_{i-1}(c - c_i, k - 1) + z_i\}, \tag{1}$$

$$i = 1, \dots, m, c = 1, \dots, C_j, k = 1, \dots, n_{\max}$$

$$U_{\max} = \max_n f_m(C, n)$$

c

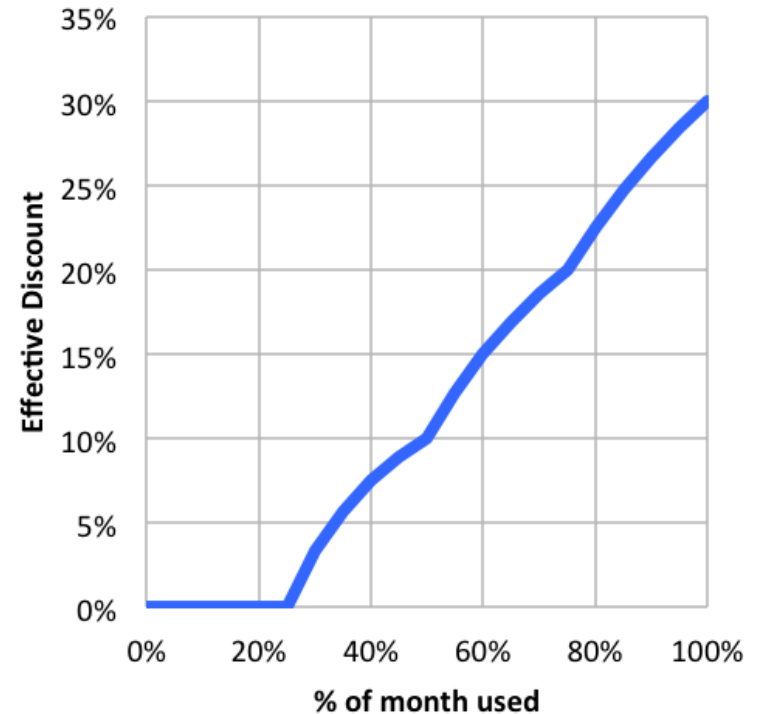
i

k=1	c=3 z=3	c=2 z=1	c=4 z=6
0	-	-	-
1	-	-	-
2	-	1	1
3	3	3	3
4	3	3	6
5	3	3	6
6	3	3	6

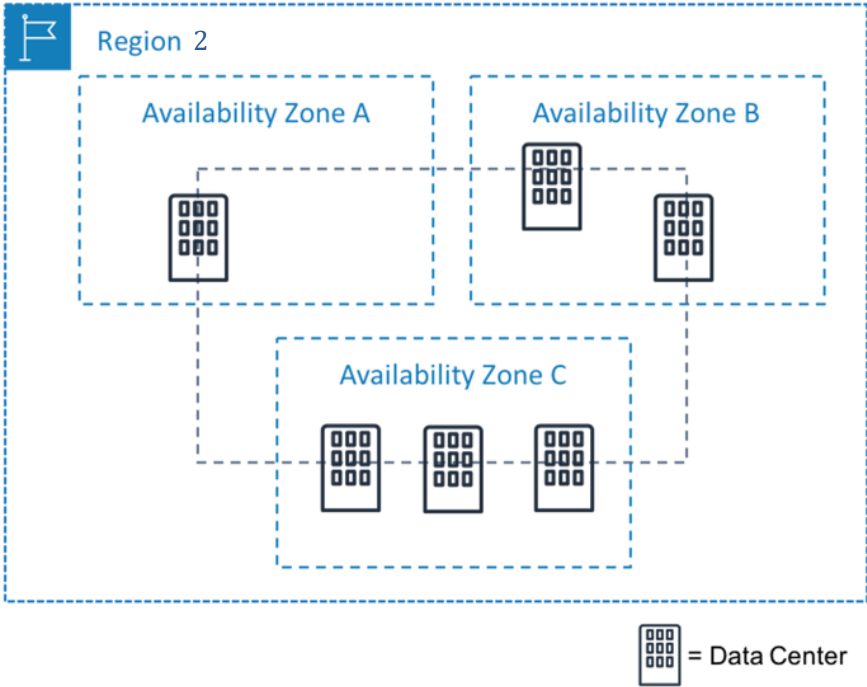
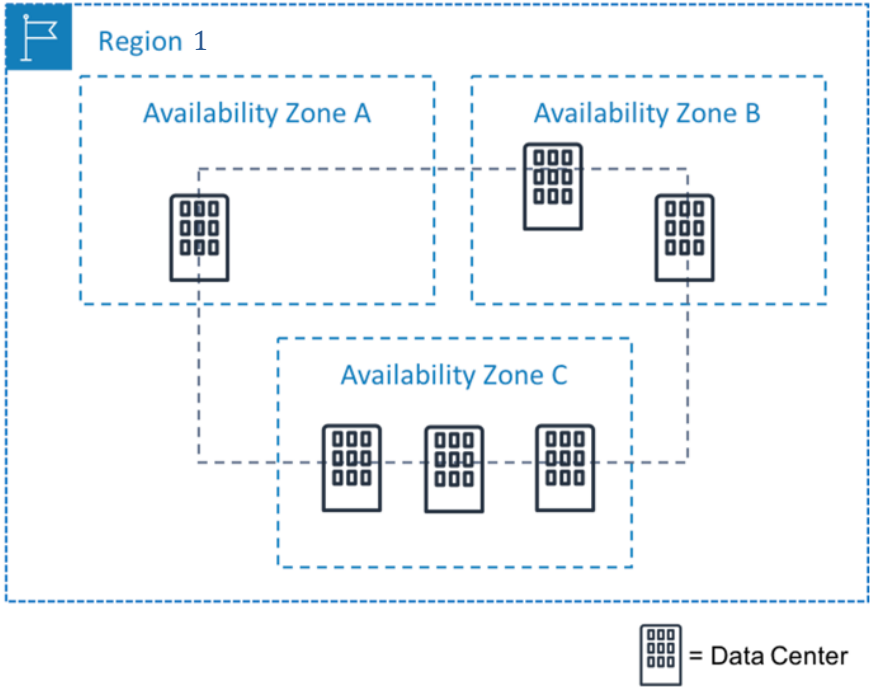
k=2	c=3 z=3	c=2 z=1	c=4 z=6
0	-	-	-
1	-	-	-
2	-	-	-
3	-	-	-
4	-	-	-
5	-	4	4
6	-	4	7

$$O(m * n_{\max} * C)$$

- Volume discount: resource provider offers a reduced price for a larger quantity of services or resources
 - Google sustained use discounts
 - Yandex committed volume of services
 - Amazon S3 data transfer
- Google Cloud network services rank egress and ingress traffic in the following order:
 - Intra-Zone Connectivity
 - Inter-Zone Connectivity
 - Inter-Region Connectivity



The difference in bandwidth between the different zones can reach up to several times



We describe and consider aggregated price and performance/bandwidth benefits in terms of **group (and hierarchical) dependencies**

- The set R of cloud resources consists of multiple VM instances possibly available from different datacenters and resource providers
- Each group $G_i \in G$ is a subset of resources $r_j \in R$ with a common group dependency expressed as a special rule for aggregate cost and utility values
 - we assume that in a general case the *aggregate* cost and utility of the selected resources may differ from their *total sum* of cost and utility;
 - quantity and volume discounts, connectivity benefits, performance bonuses.
- Given the set R of VM resources and set G of non-intersecting resource groups defined over R , select a subset of $[n_{\min}; n_{\max}]$ of resources with the aggregate cost $C_a < C_{\max}$ while optimizing the aggregate utility value $U_a \rightarrow \max$.

- The **main idea** of our approach is to solve resources selection problems for each group independently and combine their results in a higher-level recurrent solution
- For each group we calculate a **Pareto-optimal set of possible allocation variants**
- Each allocation variant describes one possible subset of group resources which provides aggregate utility u for an aggregate cost C :

$$Var_i = \{n_i, C_i, u_i\}$$

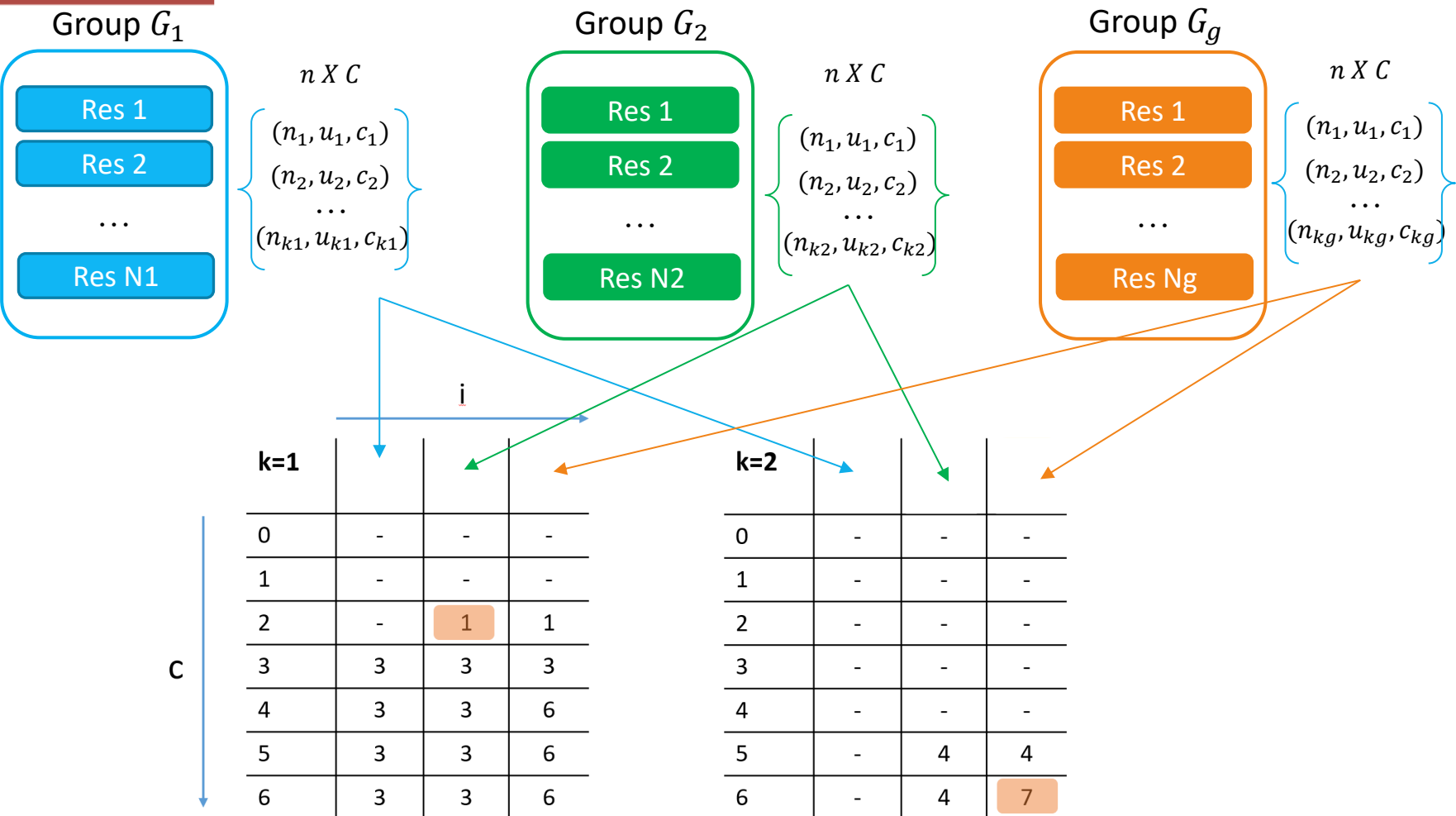
- Pareto-optimal set of variants for each subgroup can be obtained by knapsack algorithms, greedy algorithms, heuristics or brute force enumeration

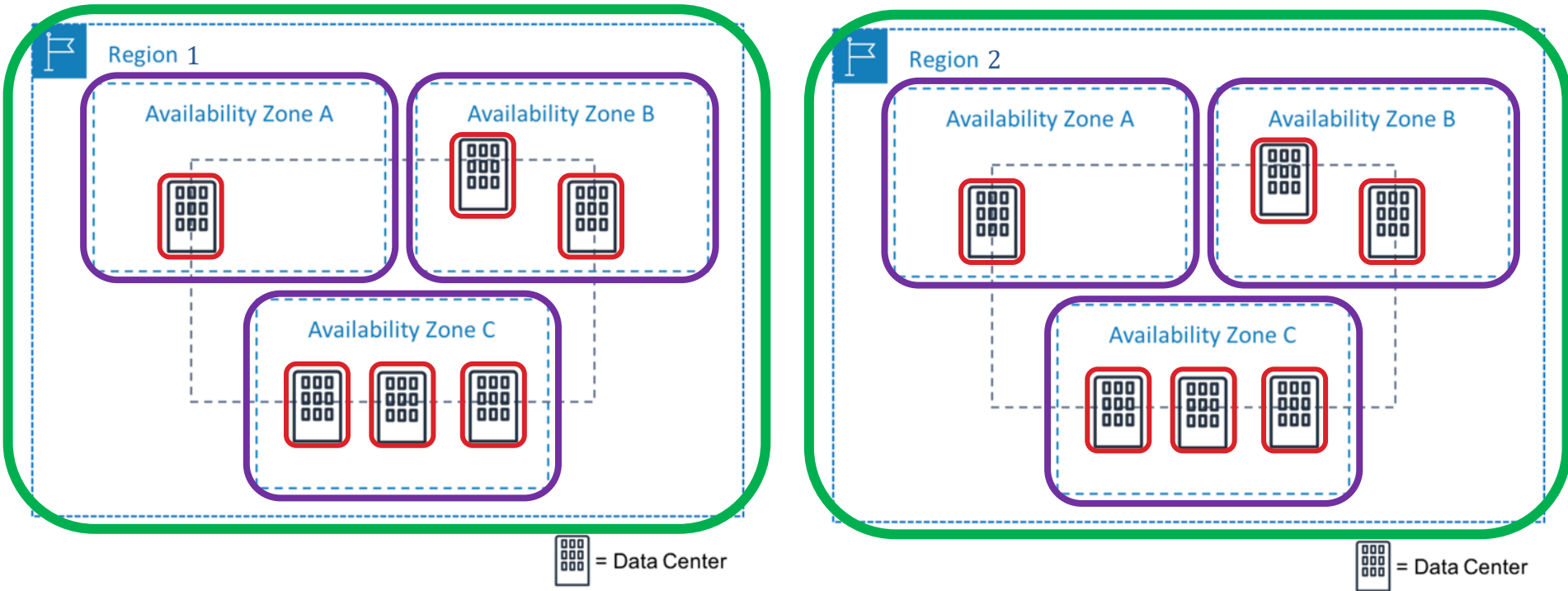
- **GKA considers groups of resources G_i as enumeration items instead of individual VMs**
- Instead of a single pair of characteristics u_i and c_i , each group item G_i provides a list of NV_i possible resource allocation *variants* $Var_j = (n_j, u_j, c_j)$
- GKA iterates over groups $G_i \in G$ and their variants $\{Var_j\}$ to calculate the following recurrent scheme:

$$f_i(c, n) = \max\{f_{i-1}(c, n), f_{i-1}(c - c_j, n - n_j) + u_j\}, \quad (2)$$

$$i = 1, \dots, |G|, j = 1, \dots, NV_i, c = 1, \dots, C_{\max}, n = 1, \dots, n_{\max}$$

- $f_i(c, n)$ then maintains the maximum possible *aggregate* utility U achievable for a subset of n VMs combined from different variants from groups $\{G_1, \dots, G_i\}$ for a budget c
- Estimated computational complexity is bounded by $O(N * n_{\max} * C_{\max}^2)$

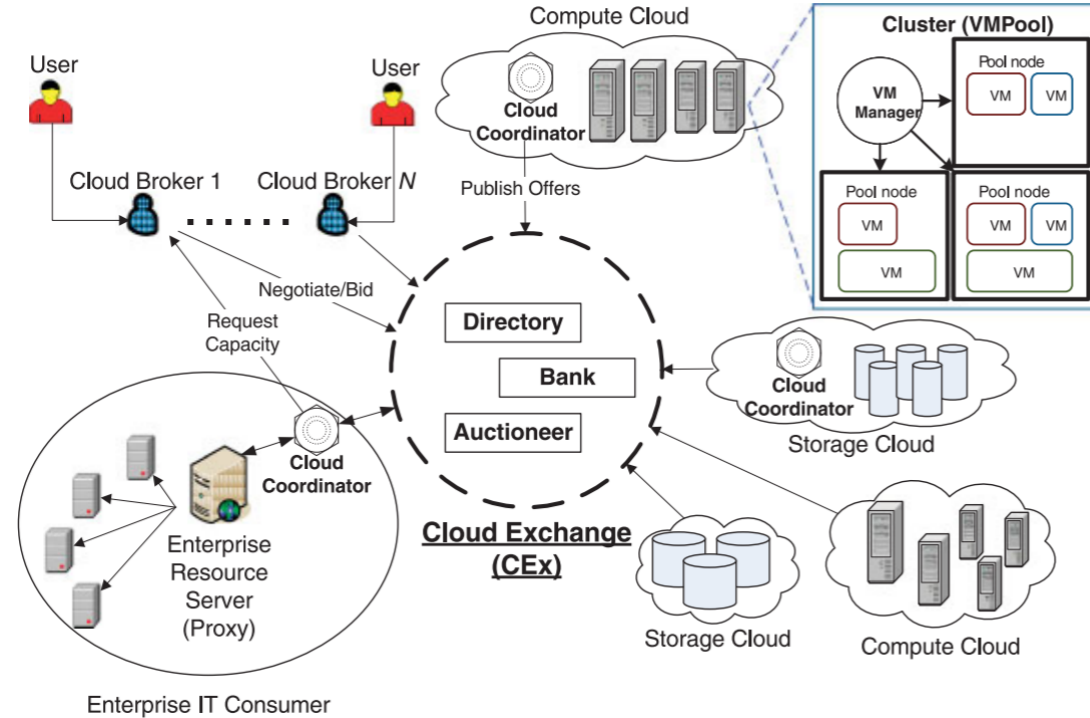




- For a more formal definition of the **hierarchical grouping** of resources, we assume a group G_i can contain other groups: $G_j^i \in G_i$.
- The hierarchical structure implies that each group G_j^i can belong to only one parent group G_i and is entirely contained within it.
- The main idea behind the hierarchical knapsack problem solution is to **recursively apply (2) across all levels of the hierarchy**.
- The solution at a lower level is passed upward as a **table of Pareto-optimal variants** $Var_j \in \{(n, p, c)\}, n = 1, \dots, n_{\max}, c = 1, \dots, C_{\max}$, where p denotes the maximum performance achievable for a given number of items n for a total cost c .

The simulation study is implemented in CloudSim v6 environment to comply with modern cloud resource provisioning model Physical resources

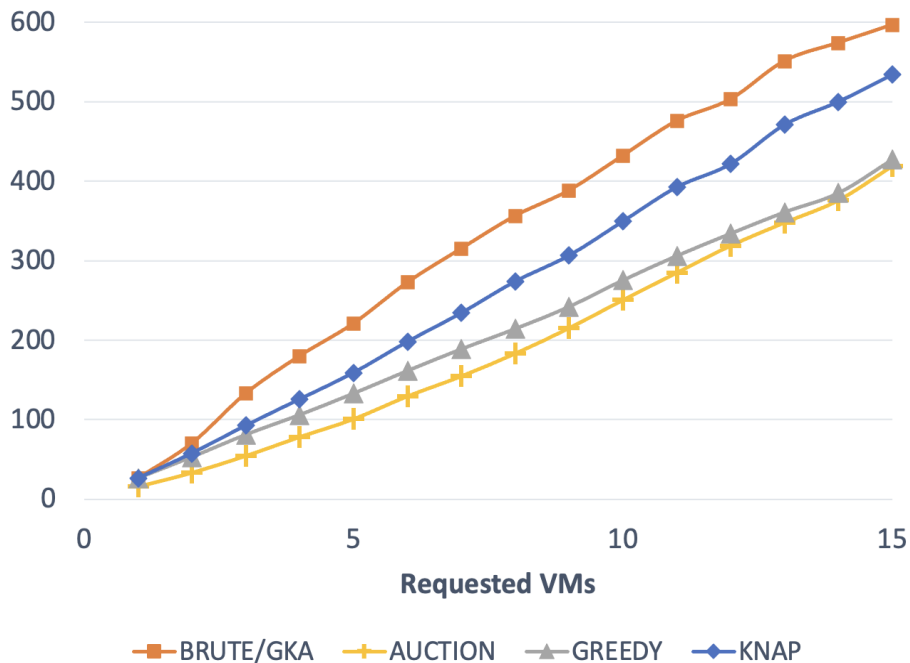
- Virtual resources and characteristics
- Datacenters
- Users and cloud brokers
- Different VM allocation policies
- Pricing models
- Event-based simulation
- Extensions
 - CloudAuction, pricing models



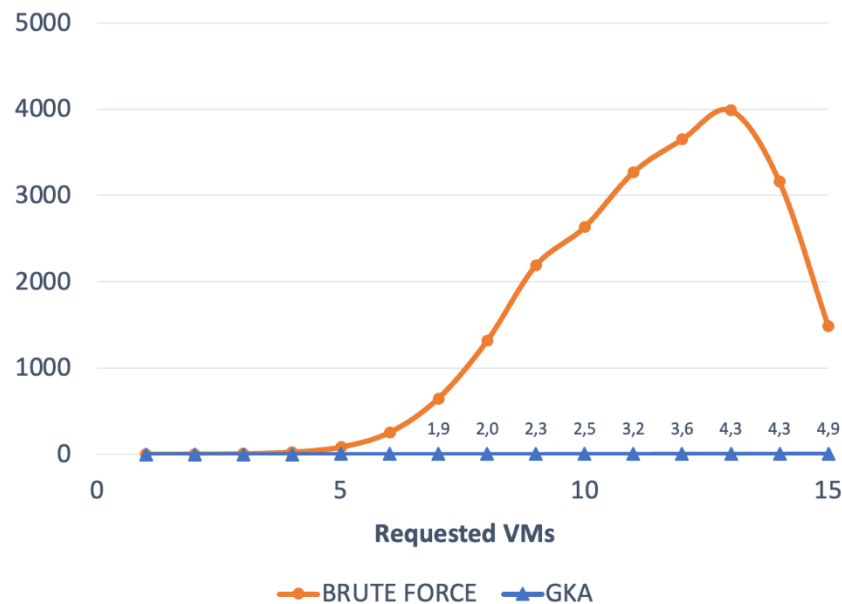
- Up to 1000 VM nodes and their characteristics (incl. bandwidth, price, RAM, etc.) are generated randomly in each simulation cycle in accordance with pre-defined tier levels
- We consider the following types of the grouping rules:
 - 1) groups without any quantity advantages or discounts
 - 2) groups with significant quantity discounts for up to 30%
 - 3) groups with significant bandwidth bonuses for up to 100%
 - 4) groups with both quantity discounts up to 20% and bandwidth bonuses up to 50%
 - For example, for a group of 10 VMs we model quantity discount as 10% for 2-4 items, 20% for 5-7 items and 30% for 8-10 items purchased
- Types of grouping rules are uniformly assigned and are equally represented in the simulated cloud environment (there are usually 67 groups over 1000 VMs)

- *Group Knapsack Algorithm (GKA)* implements proposed approach (2) to optimize VMs selection considering the resource groupings
- *Brute Force* explores all feasible combinations of available VMs and selects the best combination considering the resource groupings
- *Branch-and-Bound (BB)* algorithm provides an exact solution to the problem by systematically exploring the solution tree considering lower estimations
- *0-1 Knapsack (Knapsack)* implements VMs selection procedure (1) without information about the resource groups; group bonuses and discounts are applied to the resulting selection
- *Greedy* algorithm implements approximation of 0-1 Knapsack problem above; does not consider resource groupings; group bonuses and discounts are applied to the resulting selection

Aggregate Bandwidth, Gbps



Working Time, ms

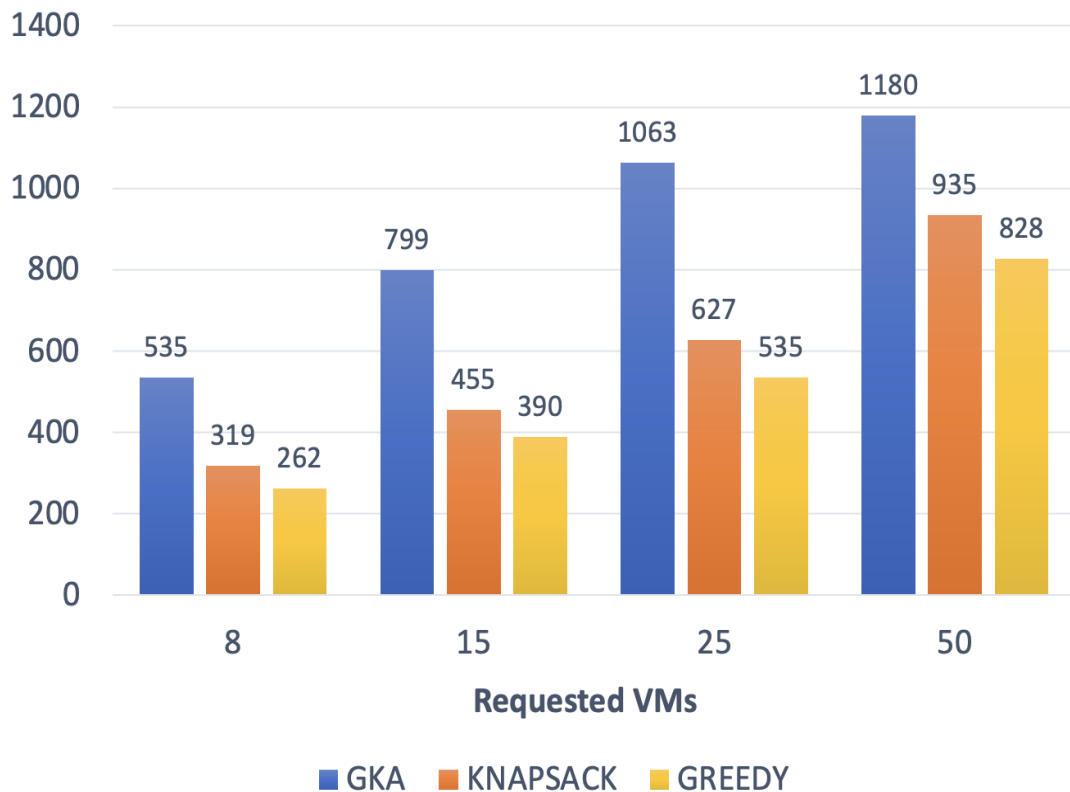


- *Brute Force* and *GKA* showed the identical best optimization results in all simulation runs, but ***Brute Force* required 1000 times more working time** for $N = 25$ VMs

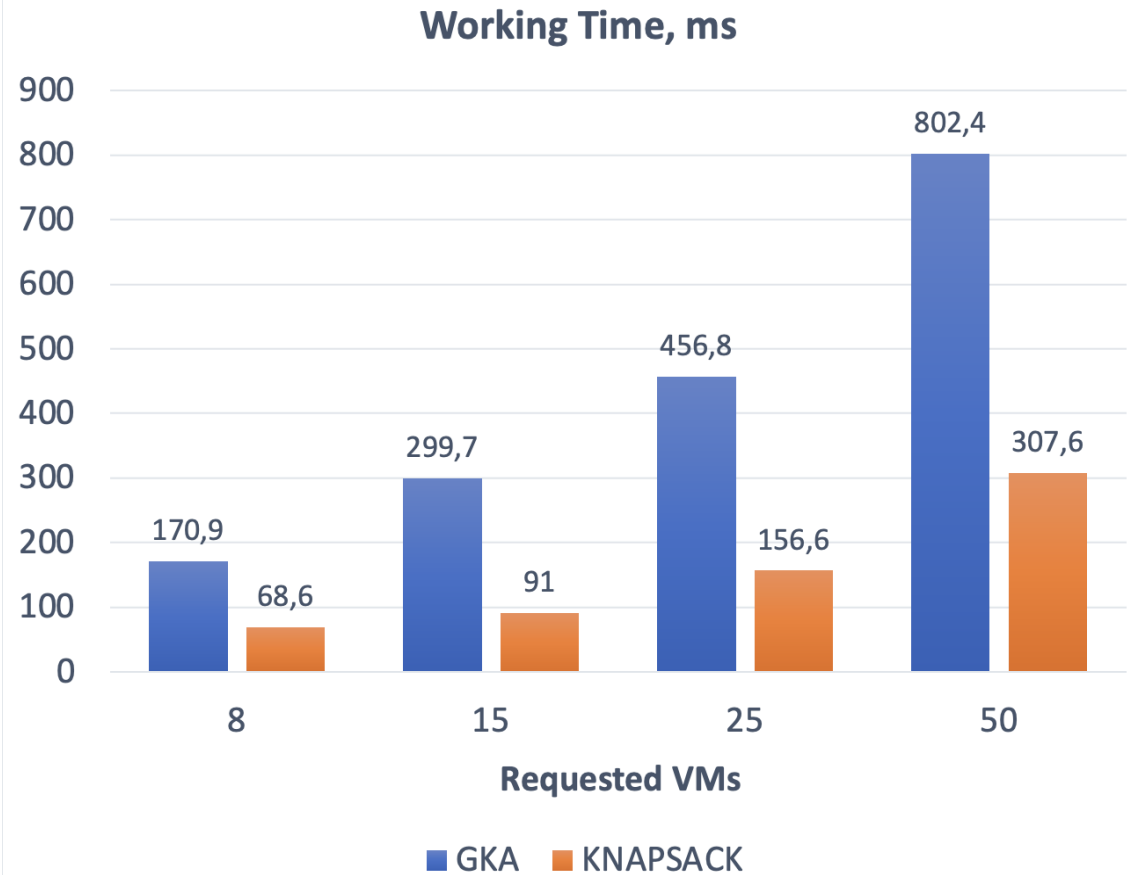
n	BB	GKA	$Knapsack$
1	1	1	1
5	5	1	1
10	31	1	1
15	90	3	3
20	206	5	5
25	506	8	7
30	1025	12	10

- *Branch-and-Bounds (BB)* and *GKA* provide **exact solution** for the problem, therefore we compare algorithms' **working time** in milliseconds
- BB is **100 times slower** in an environment with $N = 200$ VMs

Aggregate Bandwidth, Gbps



- With $n=50$ required VMs, GKA took advantage of 5% cost discounts, as well as 47% bandwidth bonuses from the resource groupings
- The **advantage of GKA** over Knapsack by the target criterion **reaches 40%**



GKA shows constant nearly **3-x higher working time** compared to classical Knapsack

- We studied the problem of efficient selection of cloud network resources considering localized, hierarchical and regional group relations and dependencies
- CloudSim package was used to simulate cloud environments with up to 30% quantity discounts and 100% bandwidth bonuses when selecting VMs from a single region group
- The proposed Group Knapsack algorithm provides accurate problem solutions identical to the brute force, while the advantage over other algorithms exceeds 40% by the target optimization criterion of total bandwidth

This work is supported by the Russian Science Foundation
(project No. 25-21-00656)

Thank you for attention!

Russian Supercomputing Days

SEPTEMBER 28 - 29, 2026