Efficient Multi-Resource, Multi-Unit VCG Auction

Liran Funaro Orna Agmon Ben-Yehuda Assaf Schuster

Department of Computer Science

TECHNION Israel Institute of Technology

International Conference on the Economic of Grids, Clouds, System, and Services (GECON)
September 17-19, 2019, Leeds, UK

Resource-as-a-Service (RaaS)

- The Resource-as-a-Service (RaaS) cloud is an elastic cloud model that allows clients to rent adjustable quantities of individual resources for short time intervals
- It characterized by:
  - Fine resource granularity
  - Fine time granularity
  - Market-driven resource pricing

More details in:

- Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, et al. "The Resource-as-a-Service (RaaS) Cloud". In: *Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*. Boston, MA: USENIX Association, 2012
- Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, et al. "The Rise of RaaS: The Resource-as-a-Service Cloud". In: *Communications of the ACM* 57.7 (2014), pp. 76–84. ISSN: 0001-0782

Resource Allocation via Auctions

- Infrastructure-as-a-Service (IaaS) providers have been using auctions to control congestion via preemptible virtual-machine (VM) instances for nearly a decade
- A natural extension of this idea is to auction additional individual resources in an existing VM
- VCG are appealing for this purpose, as they are
  - **Truthful**: they incentivize clients to reveal their true valuation of the resources, which helps cloud providers accurately price their services
  - Maximizing the **social welfare**: the aggregate valuation the clients assign to the chosen resource allocation

More details in:

- Orna Agmon Ben-Yehuda, Eyal Posener, et al. "Ginseng: Market-driven Memory Allocation". In: *Proceedings of the 10th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE)*. vol. 49. 7. Salt Lake City, Utah, USA: ACM, 2014. ISBN: 978-1-4503-2764-0
- Liran Funaro, Orna Agmon Ben-Yehuda, and Assaf Schuster. "Ginseng: market-driven LLC allocation". In: *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference*. USENIX Association. ACM, 2016, pp. 295–308

Resource Auction Protocol

Each client rents a base resource bundle that is reserved for that client, then...

Resource Auction Protocol

Each client rents a base resource bundle that is reserved for that client, then...

The host announces an auction every few seconds

Resource Auction Protocol

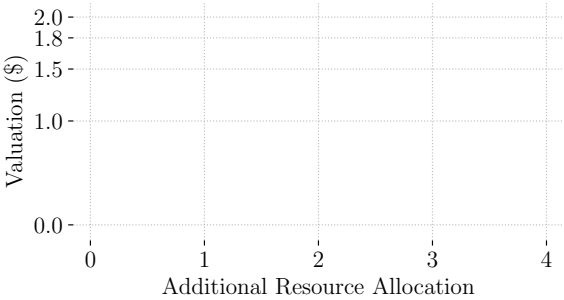
Each client rents a base resource bundle that is reserved for that client, then...

The host announces an auction every few seconds

Each agent, representing the client, bids with a valuation for each quantity of additional bundle of resources — how much it is worth, subjectively

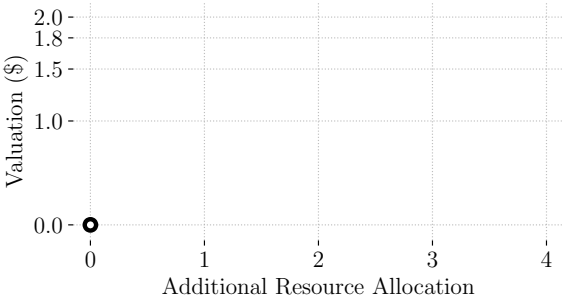
Client's Valuation Function

- A client evaluates its benefit from each additional resource bundle



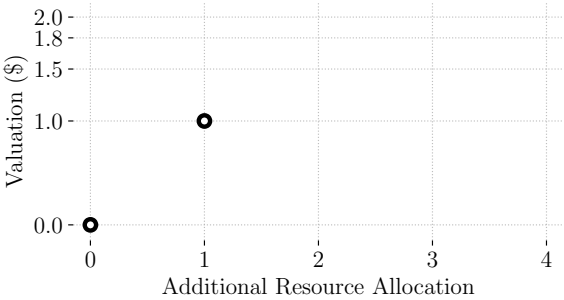
Client's Valuation Function

- A client evaluates its benefit from each additional resource bundle



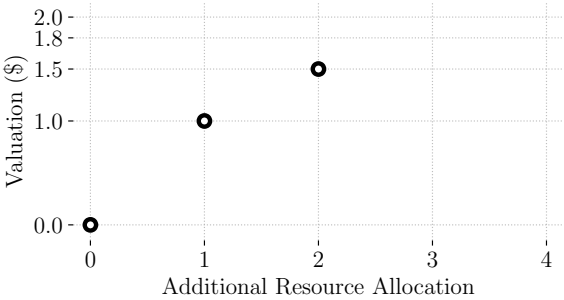
Client's Valuation Function

- A client evaluates its benefit from each additional resource bundle



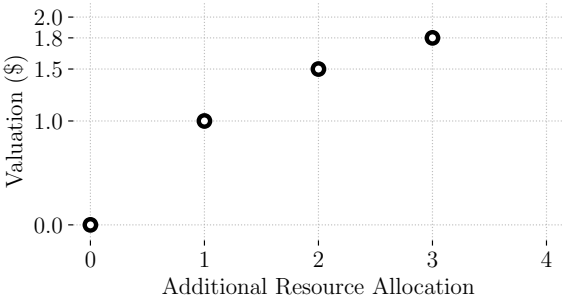
Client's Valuation Function

- A client evaluates its benefit from each additional resource bundle



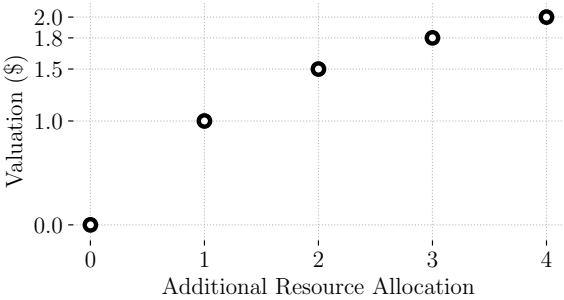
Client's Valuation Function

- A client evaluates its benefit from each additional resource bundle



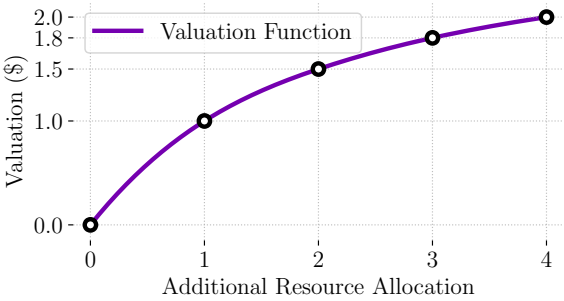
Client's Valuation Function

- A client evaluates its benefit from each additional resource bundle



Client's Valuation Function

- A client evaluates its benefit from each additional resource bundle
- Its agent then submit all these valuations as its **valuation function**



Resource Auction Protocol

Each client rents a base resource bundle that is reserved for that client, then...

The host announces an auction every few seconds

Each agent, representing the client, bids with a valuation for each quantity of additional bundle of resources — how much it is worth, subjectively

The host solves an optimization problem: finding the allocation that maximize the social welfare

Resource Auction Protocol

Each client rents a base resource bundle that is reserved for that client, then...

The host announces an auction every few seconds

Each agent, representing the client, bids with a valuation for each quantity of additional bundle of resources — how much it is worth, subjectively

The host solves an optimization problem: finding the allocation that maximize the social welfare

The host informs the agents of their allocation and charges them according to the **exclusion-compensation** principle

Exclusion-Compensation Principle

- **Exclusion-Compensation Principle**: Each guest pays for the damage it inflicted on the other guests in the system
- The auctioneer must determine the social welfare that could be achieved when that winning agent is excluded from the auction

Exclusion-Compensation Principle

- **Exclusion-Compensation Principle**: Each guest pays for the damage it inflicted on the other guests in the system
- The auctioneer must determine the social welfare that could be achieved when that winning agent is excluded from the auction

The good:

- The guests are **truthful**: they cannot improve their status by bidding a higher or a lower value
- If the demand is low, clients can rent the additional resources for a very low price, which is financially beneficial to the clients

Exclusion-Compensation Principle

- **Exclusion-Compensation Principle**: Each guest pays for the damage it inflicted on the other guests in the system
- The auctioneer must determine the social welfare that could be achieved when that winning agent is excluded from the auction

The good:

- The guests are **truthful**: they cannot improve their status by bidding a higher or a lower value
- If the demand is low, clients can rent the additional resources for a very low price, which is financially beneficial to the clients

The bad:

- The provider needs to repeat the optimization for each winning client, without its participation in the auction

The Problem: Complexity

- Finding the optimal allocation has a high computational complexity

Existing solutions:

The Problem: Complexity

- Finding the optimal allocation has a high computational complexity

Existing solutions:

- **Relaxing** the problem's definition, e.g., only concave valuation functions (Lazar and Semret 1999; Maillé and Tuffin 2004; Bae et al. 2008; Agmon Ben-Yehuda, Posener, et al. 2014)
  - Unrealistic (Funaro, Agmon Ben-Yehuda, and Schuster 2019b; Ye et al. 2015; Cameron and Singer 2014)
  - Known solutions only applicable to a single resource

The Problem: Complexity

- Finding the optimal allocation has a high computational complexity

Existing solutions:

- **Relaxing** the problem's definition, e.g., only concave valuation functions
  (Lazar and Semret 1999; Maillé and Tuffin 2004; Bae et al. 2008; Agmon Ben-Yehuda, Posener, et al. 2014)
  - Unrealistic (Funaro, Agmon Ben-Yehuda, and Schuster 2019b; Ye et al. 2015; Cameron and Singer 2014)
  - Known solutions only applicable to a single resource
- **Approximation**: non-optimal social welfare
  (Gao et al. 2017; Akbar et al. 2006; Moser, Jokanovic, and Shiratori 1997; Khan et al. 2002; Hifi, Michrafy, and Sbihi 2004)
  - Reduces the client's incentives to be truthful
  - In turn further deflects from the optimal social welfare

The Problem: Complexity

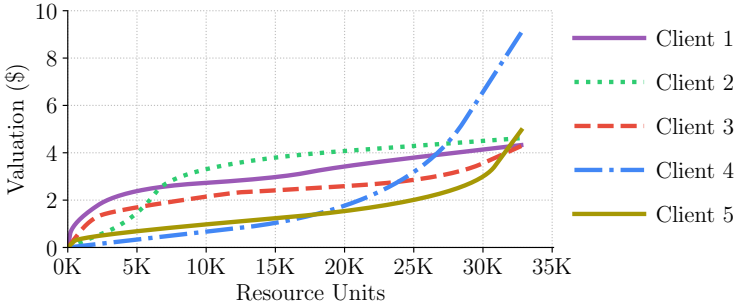
- Finding the optimal allocation has a high computational complexity

Existing solutions:

- **Relaxing** the problem's definition, e.g., only concave valuation functions
  (Lazar and Semret 1999; Maillé and Tuffin 2004; Bae et al. 2008; Agmon Ben-Yehuda, Posener, et al. 2014)
  - Unrealistic (Funaro, Agmon Ben-Yehuda, and Schuster 2019b; Ye et al. 2015; Cameron and Singer 2014)
  - Known solutions only applicable to a single resource
- **Approximation**: non-optimal social welfare
  (Gao et al. 2017; Akbar et al. 2006; Moser, Jokanovic, and Shiratori 1997; Khan et al. 2002; Hifi, Michrafy, and Sbihi 2004)
  - Reduces the client's incentives to be truthful
  - In turn further deflects from the optimal social welfare
- **B&B**: branch-and-bound algorithms
  (Ghassemi-Tari, Hendizadeh, and Hogg 2018; Hifi, Sadfi, and Sbihi 2004; Sbihi 2007; Razzazi and Ghasemi 2008; Gonen and Lehmann 2000)
  - Only tested empirically with small datasets
  - Did not scale well for many clients and large, complete valuation functions
    - Nonpolynomial increase in runtime with respect to the number of possible allocations

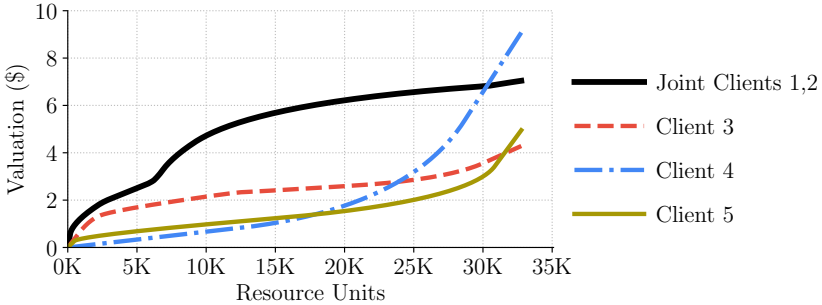
Joint Valuation Algorithm

- **Dynamic Programming**: joint-valuation algorithm
  Liran Funaro, Orna Agmon Ben-Yehuda, and Assaf Schuster. "Ginseng: market-driven LLC allocation". In: *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference*. USENIX Association. ACM, 2016, pp. 295–308
- We have five clients



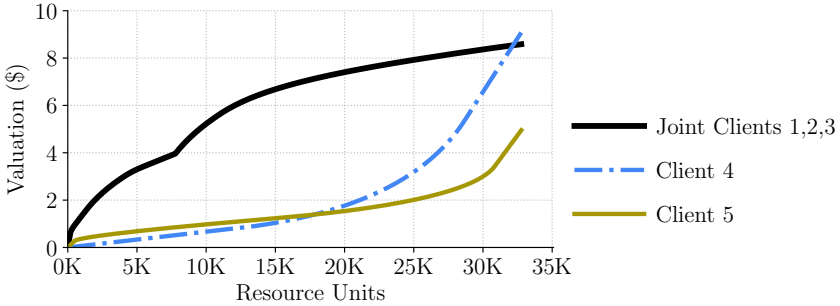
Joint Valuation Algorithm

- **Dynamic Programming**: joint-valuation algorithm
  Liran Funaro, Orna Agmon Ben-Yehuda, and Assaf Schuster. "Ginseng: market-driven LLC allocation". In: *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference*. USENIX Association. ACM, 2016, pp. 295–308
- We have five clients



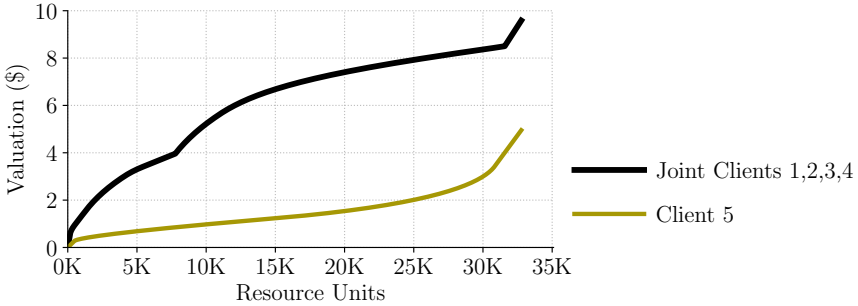
Joint Valuation Algorithm

- **Dynamic Programming**: joint-valuation algorithm
  Liran Funaro, Orna Agmon Ben-Yehuda, and Assaf Schuster. "Ginseng: market-driven LLC allocation". In: *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference*. USENIX Association. ACM, 2016, pp. 295–308
- We have five clients



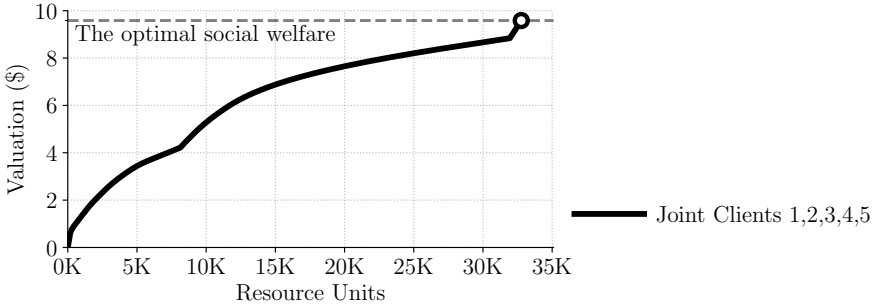
Joint Valuation Algorithm

- **Dynamic Programming**: joint-valuation algorithm
  Liran Funaro, Orna Agmon Ben-Yehuda, and Assaf Schuster. "Ginseng: market-driven LLC allocation". In: *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference*. USENIX Association. ACM, 2016, pp. 295–308
- We have five clients



Joint Valuation Algorithm

- **Dynamic Programming**: joint-valuation algorithm
  Liran Funaro, Orna Agmon Ben-Yehuda, and Assaf Schuster. "Ginseng: market-driven LLC allocation". In: *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference*. USENIX Association. ACM, 2016, pp. 295–308
- We have five clients

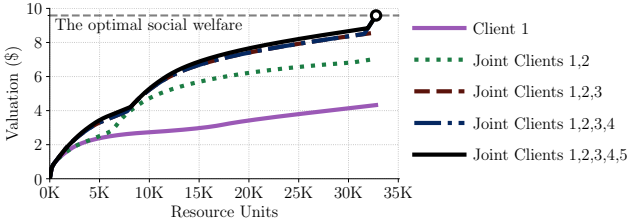


Joint Valuation Algorithm Complexity

- For each joining of two valuations, we need to compare all the allocation combinations between the two clients
  - Time complexity: $O(n \cdot N^2)$
    - n: the number of clients
    - N: the number of possible unit allocations for each client
- Reasonable for single resource
- For four resources, each with 15 units and 256 clients, this can add up to a full hour (excluding payment calculation)
  - E.g., RAM, CPU, BW and LLC
- Forces a long time period between auctions
  - Infeasible in a RaaS system which requires fine time granularity

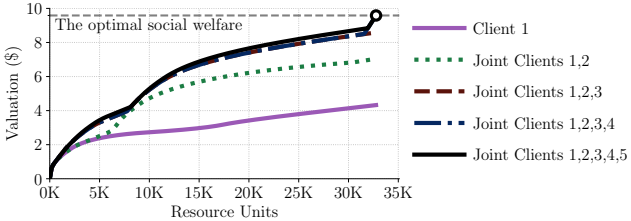
First Improvement: Payment Calculation

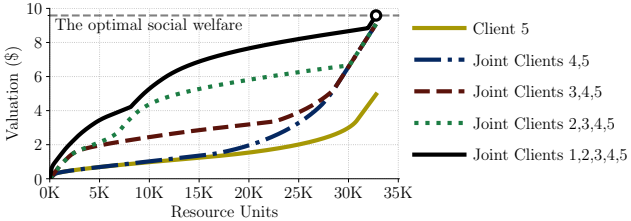
- We demonstrated joining the clients in one order



First Improvement: Payment Calculation

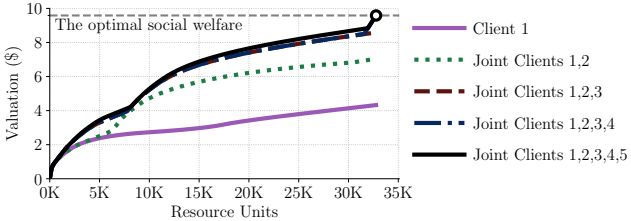
- We demonstrated joining the clients in one order
- We can join them also in the reverse order

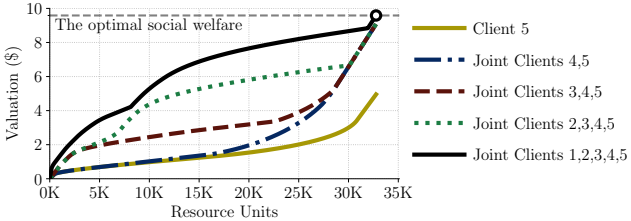




First Improvement: Payment Calculation (2)

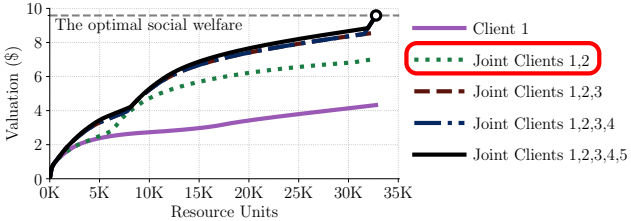
- Let's say we want to calculate the optimal social welfare without client 3

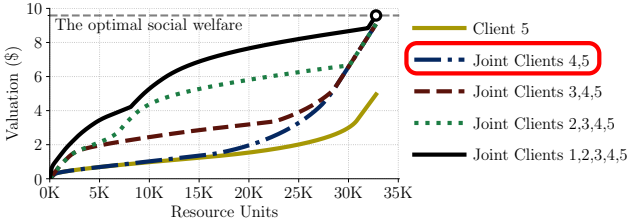




First Improvement: Payment Calculation (2)

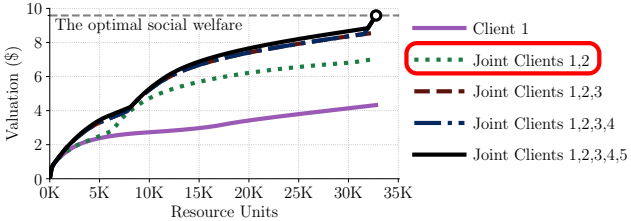
- Let's say we want to calculate the optimal social welfare without client 3
- We can take the joint valuation of clients 1,2 and clients 4,5 from the reverse order

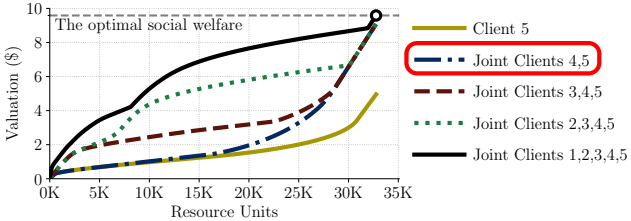


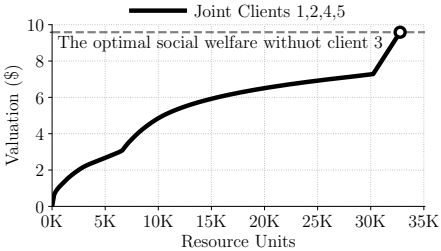


First Improvement: Payment Calculation (2)

- Let's say we want to calculate the optimal social welfare without client 3
- We can take the joint valuation of clients 1,2 and clients 4,5 from the reverse order







Auction Complexity

- Now we can calculate the payments in $O(w \cdot N^2)$
  - Instead of $O(w \cdot n \cdot N^2)$
  - w: The number of winning clients

Auction Complexity

- Now we can calculate the payments in $O(w \cdot N^2)$
  - Instead of $O(w \cdot n \cdot N^2)$
  - w: The number of winning clients
- But it could still take over an hour to calculate the optimal allocation

Auction Complexity

- Now we can calculate the payments in $O(w \cdot N^2)$
  - Instead of $O(w \cdot n \cdot N^2)$
  - w: The number of winning clients
- But it could still take over an hour to calculate the optimal allocation
- We can reduce the complexity of joining two valuations by reducing the number of compared allocations.
- To do so, we filter out allocations that cannot maximize the social welfare.

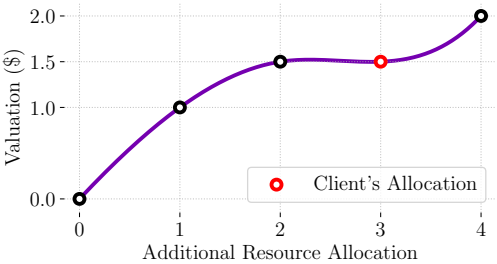
Auction Complexity

- Now we can calculate the payments in $O(w \cdot N^2)$
  - Instead of $O(w \cdot n \cdot N^2)$
  - w: The number of winning clients
- But it could still take over an hour to calculate the optimal allocation
- We can reduce the complexity of joining two valuations by reducing the number of compared allocations.
- To do so, we filter out allocations that cannot maximize the social welfare.
- If an allocation globally maximizes the social welfare, then it is
  1. **Pareto efficient**: one agent's allocation cannot be improved without hindering another's
  2. A **local optimum**: the aggregated valuation cannot be increased by taking a resource from one agent and giving it to another.

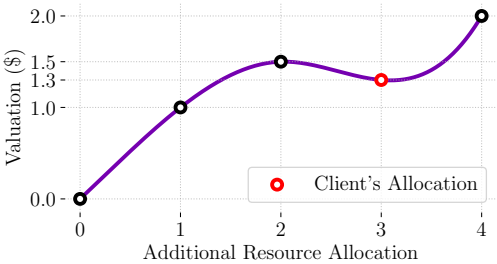
Pareto Efficient Allocation

- **If the allocation is optimal**, no client can be **indifferent** or **benefit** if we reduce its allocation

Indifferent to allocation reduction:



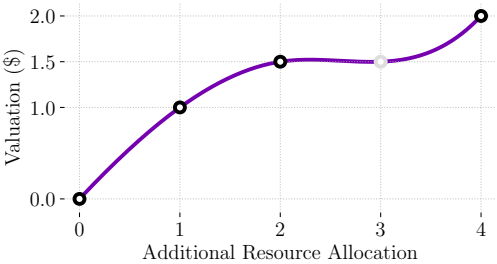
Benefit from allocation reduction:



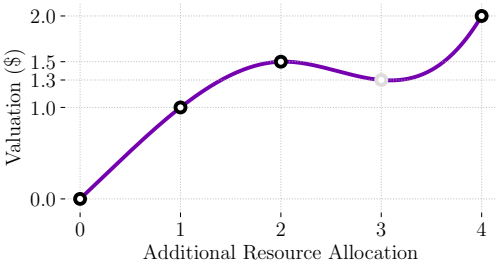
Pareto Efficient Allocation

- **If the allocation is optimal**, no client can be **indifferent** or **benefit** if we reduce its allocation
  - We remove such allocations from the client's valuation

Indifferent to allocation reduction:

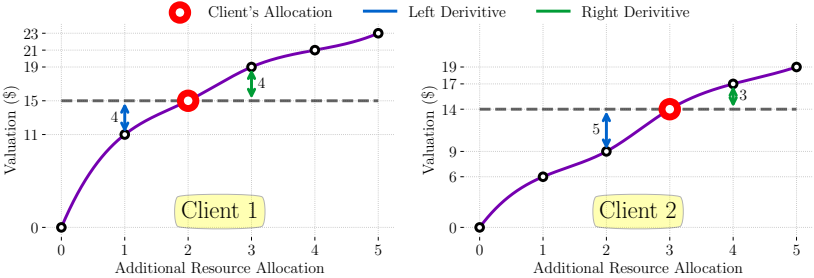


Benefit from allocation reduction:



Local Optimum

- **For an optimal allocation**, any right partial derivative of any **single** agent's valuation function is no greater than any of the **other** agents' left partial derivatives[1]

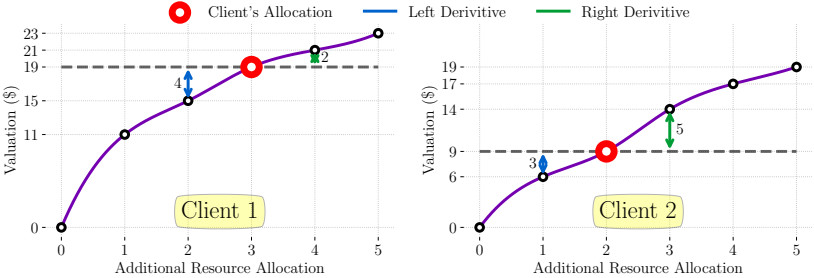


Social welfare $= 29$

[1] We define the left/right partial derivatives as the difference in the values between adjacent points in the allocation space ($dx = 1$ for all the resources)

Local Optimum

- **For an optimal allocation**, any right partial derivative of any **single** agent's valuation function is no greater than any of the **other** agents' left partial derivatives[1]

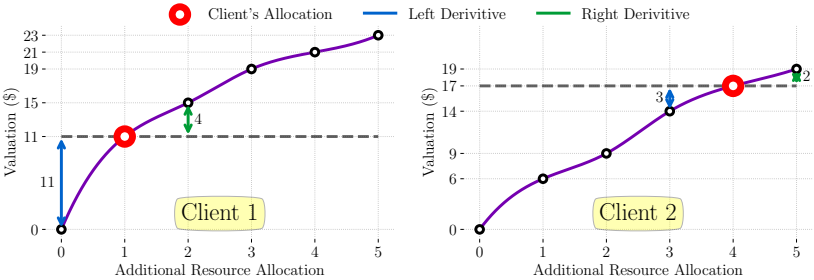


Social welfare $= 28$

[1] We define the left/right partial derivatives as the difference in the values between adjacent points in the allocation space ($dx = 1$ for all the resources)

Local Optimum

- **For an optimal allocation**, any right partial derivative of any **single** agent's valuation function is no greater than any of the **other** agents' left partial derivatives[1]

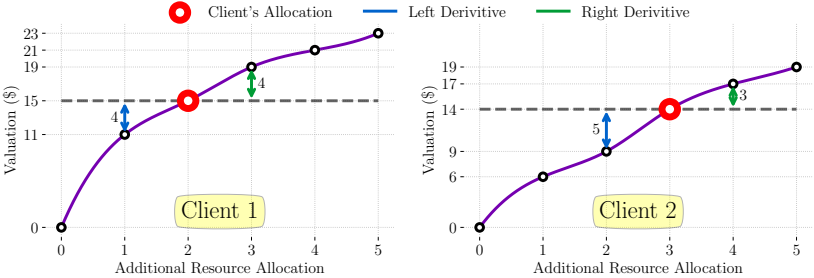


Social welfare $= 28$

[1] We define the left/right partial derivatives as the difference in the values between adjacent points in the allocation space ($dx = 1$ for all the resources)

Local Optimum

- **For an optimal allocation**, any right partial derivative of any **single** agent's valuation function is no greater than any of the **other** agents' left partial derivatives[1]
  - When joining two clients, for each allocation of one client we can compare only the allocations of the other client that fit this rule

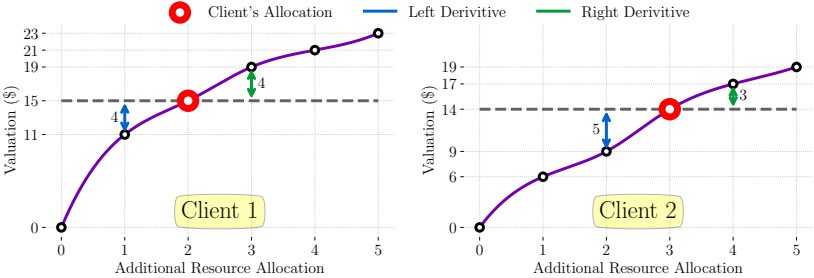


Social welfare $= 29$

[1] We define the left/right partial derivatives as the difference in the values between adjacent points in the allocation space ($dx = 1$ for all the resources)

Local Optimum

- **For an optimal allocation**, any right partial derivative of any **single** agent's valuation function is no greater than any of the **other** agents' left partial derivatives[1]
  - When joining two clients, for each allocation of one client we can compare only the allocations of the other client that fit this rule
  - This requires pre-processing



Social welfare $= 29$

[1] We define the left/right partial derivatives as the difference in the values between adjacent points in the allocation space ($dx = 1$ for all the resources)

Local Optimum Preprocess

- When joining two clients, we first pre-process the valuation function of one of the client
- Then, for each allocation of the other client, we find the points that preserve the local optimum
  - This can be done efficiently using k-dimensional upper-bound data structure
  - Its construction time complexity is $O(N \log N)$
  - Its lookup time complexity is $O(\log N)$
  - So the total complexity for comparing all of the allocations is $O(N \log N)$

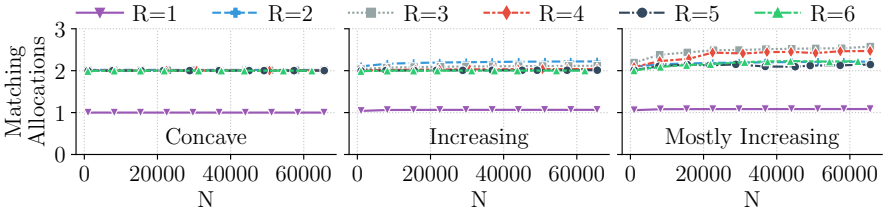
- The lookup time complexity is only true if the number of matching allocations is constant (independent of N and n) and relatively low

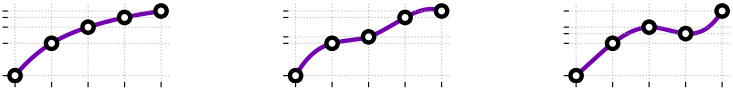
More details in:

- Liran Funaro, Orna Agmon Ben-Yehuda, and Assaf Schuster. "Efficient Multi-Resource, Multi-Unit VCG Auction". In: *arXiv e-prints*, arXiv:1905.09014 (May 2019), arXiv:1905.09014. arXiv: `1905.09014 [cs.GT]`

Local Optimum Preprocess Empirical Results

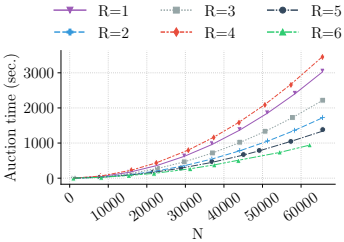
- In the paper, we prove that the average number of possible allocations that do not violate the local-optimum rule is **constant**
- We also show that empirically in our evaluation



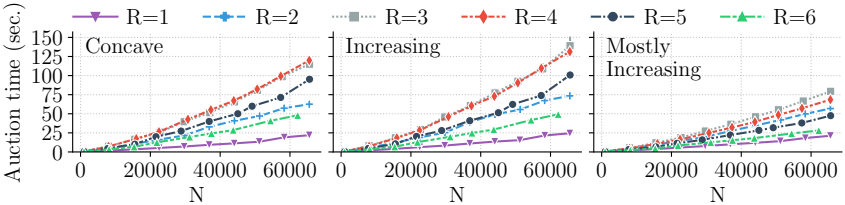


Our Improvement

Baseline joint-valuation algorithm:



Our improved joint-valuation algorithm:



Conclusions

- Our algorithm allows cloud providers to implement the RaaS model
- They can deploy a multi-resource auction for allocation of additional resources in an existing VM every two minutes for up to 256 clients in a single physical machine
  - Over 20 times improvement compared with the original algorithm
  - Without restrictions on the valuation functions
- The implementation of our algorithm is available from `https://bitbucket.org/funaro/vecfunc-vcg`

Questions?

Liran Funaro: `funaro@cs.technion.ac.il`