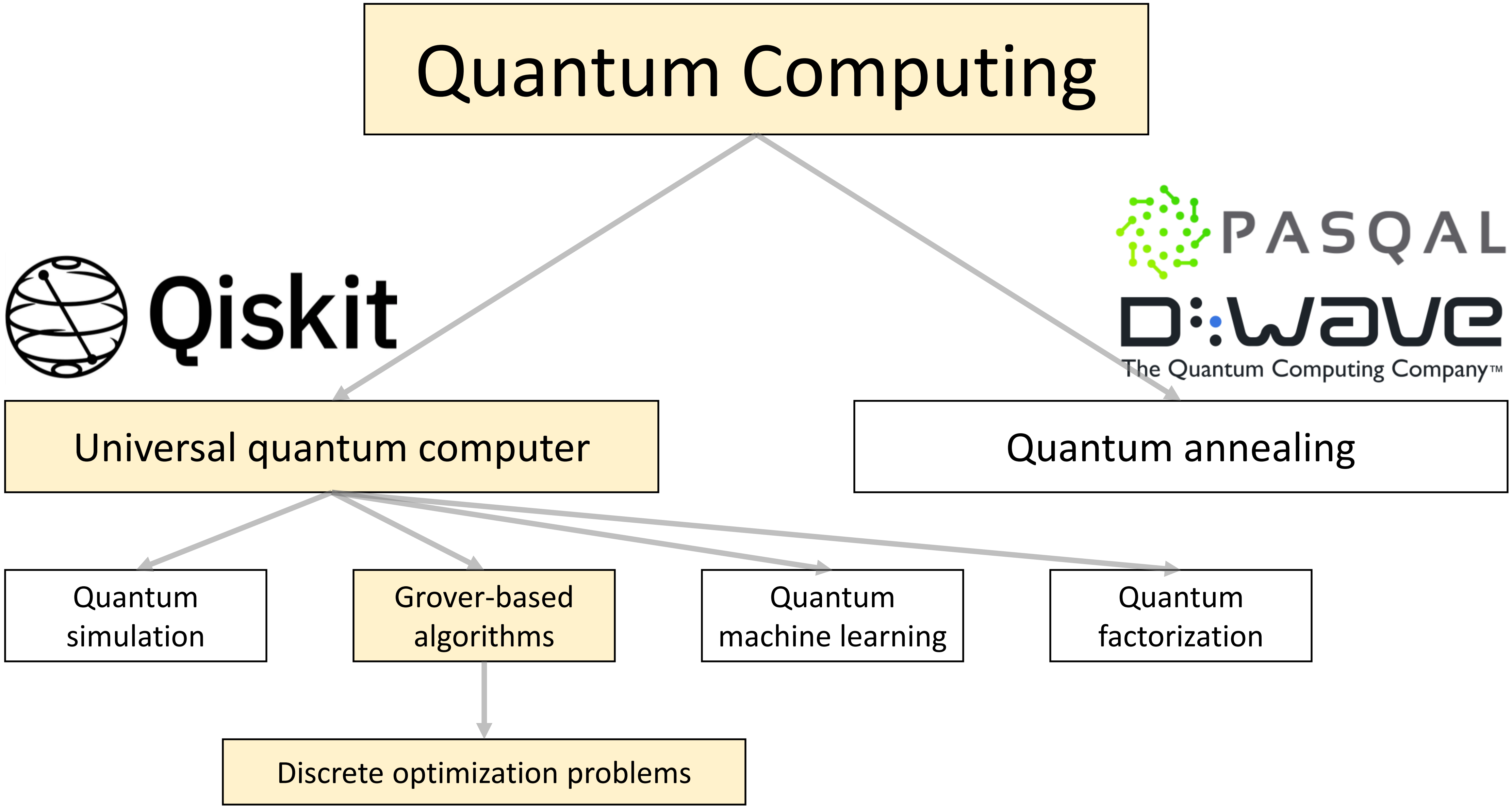
Discrete Optimization

A Quantum Revolution?

Stefan Creemers
Luis Fernando Pérez

IÉSEG SCHOOL OF MANAGEMENT

KATHOLIEKE UNIVERSITEIT LEUVEN


Quantum Computing
Qiskit
PASQAL
D:WAVE
The Quantum Computing Company™
Universal quantum computer
Quantum annealing
Quantum simulation
Grover-based algorithms
Quantum machine learning
Quantum factorization
Discrete optimization problems

Discrete optimization problems

- In the most general form:

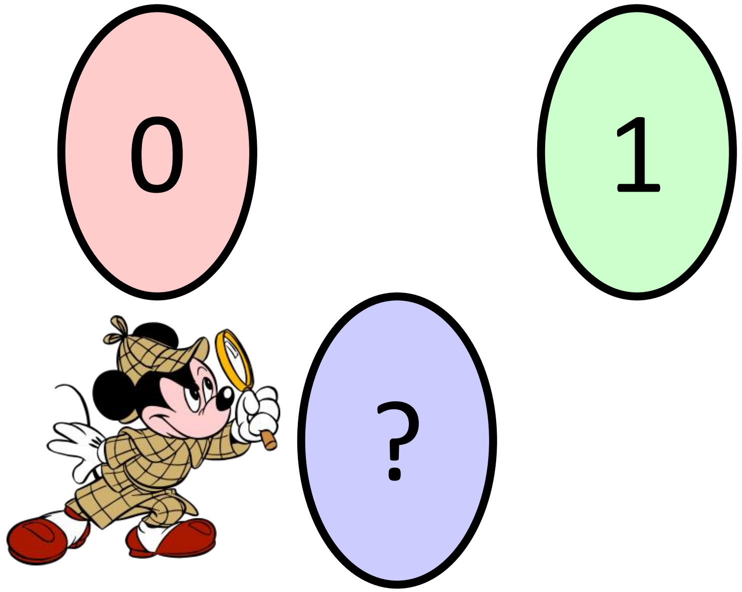
  optimize $g(x_1, x_2, \ldots, x_n)$
  subject to
  $x_i \in \Omega_i, \forall i: 0 \leq i \leq n$
  (any other constraint)

- Where:
  - $g(\boldsymbol{x})$ is the objective function that evaluates assignment $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$.
  - n is the number of decision variables.
  - x_i is the i^{th} decision variable.
  - Ω_i is the set of discrete values that can be assigned to decision variable x_i.
- Objective function and/or constraints do not have to be linear!
- Examples include: 3SAT, knapsack, TSP, complex non-linear integer programming problems, and most other OR problems discussed here at ROADEF!

Basic unit of information: Classic vs quantum

Classical computing

- Bit.
- Can take on values 0 and 1.



Quantum computing

- Qubit.
- Can take on values 0 and 1.
- Can be in a superposition state.
- Only after observing the qubit, the state collapses to basis state 0 or 1.
- The probability that the state of a qubit collapses to 0 or 1 depends on the superposition.
- In case of a uniform superposition, there is a 50% chance to collapse into either 0 or 1.

Solving the binary knapsack problem

- $n = 3$ items.
- Maximum weight $W = 4$.
- Optimal solution value $V^* = 5$.
- Solution $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$.
- Weight of $\boldsymbol{x}$ is $W_{\boldsymbol{x}}$.
- Value of $\boldsymbol{x}$ is $V_{\boldsymbol{x}}$.
- Function $f(\boldsymbol{x})$ evaluates whether solution $\boldsymbol{x}$ is valid; has weight $W_{\boldsymbol{x}}$ that does not exceed weight capacity W, and value $V_{\boldsymbol{x}}$ is at least equal to V^*.

i	w_i	v_i
1	2	3
2	3	1
3	2	2
$n = 3$	$W = 4$	$V^* = 5$
$\boldsymbol{x} = \{x_1, x_2, x_3\}$	$W_{\boldsymbol{x}} = \sum w_i x_i$	$V_{\boldsymbol{x}} = \sum v_i x_i$
$f(\boldsymbol{x}) = 1$ if $W_{\boldsymbol{x}} \leq W$ and $V_{\boldsymbol{x}} \geq V^*$		

Solving the binary knapsack problem

- Classical computing:
  - Full enumeration requires $2^n = 8$ calls to function $f(\boldsymbol{x})$.
  - Each call to $f(\boldsymbol{x})$ requires η operations.
  - In case of knapsack, $\eta = O(n)$ ➔ full enumeration has complexity $O(n2^n)$.
  - Best classical algorithm to solve binary knapsack has complexity $O(n\sqrt{2^n})$.
- Quantum computing:
  - Given a (uniform) superposition of three qubits, only a single call to $f(\boldsymbol{x})$ is required to obtain $f(\boldsymbol{x})$ for each possible solution ➔ complexity $O(n)$?
  - Each solution, however, has probability $2^{-n} = 0.125$ to be measured ➔ we only have a 12.5% chance to measure 101.

i	w_i	v_i
1	2	3
2	3	1
3	2	2
$n = 3$	$W = 4$	$V^* = 5$
$\boldsymbol{x} = \{x_1, x_2, x_3\}$	$W_{\boldsymbol{x}} = \sum w_i x_i$	$V_{\boldsymbol{x}} = \sum v_i x_i$
$f(\boldsymbol{x}) = 1$ if $W_{\boldsymbol{x}} \leq W$ and $V_{\boldsymbol{x}} \geq V^*$		

$\boldsymbol{x}$	$W_{\boldsymbol{x}}$	$V_{\boldsymbol{x}}$	$f(\boldsymbol{x})$	$P(x)$
000	0	0	0	0.125
100	2	3		0.125
010	3	1		0.125
110	5	4		0.125
001	2	2		0.125
101	4	5	1	0.125
011	5	3	0	0.125
111	7	6	0	0.125

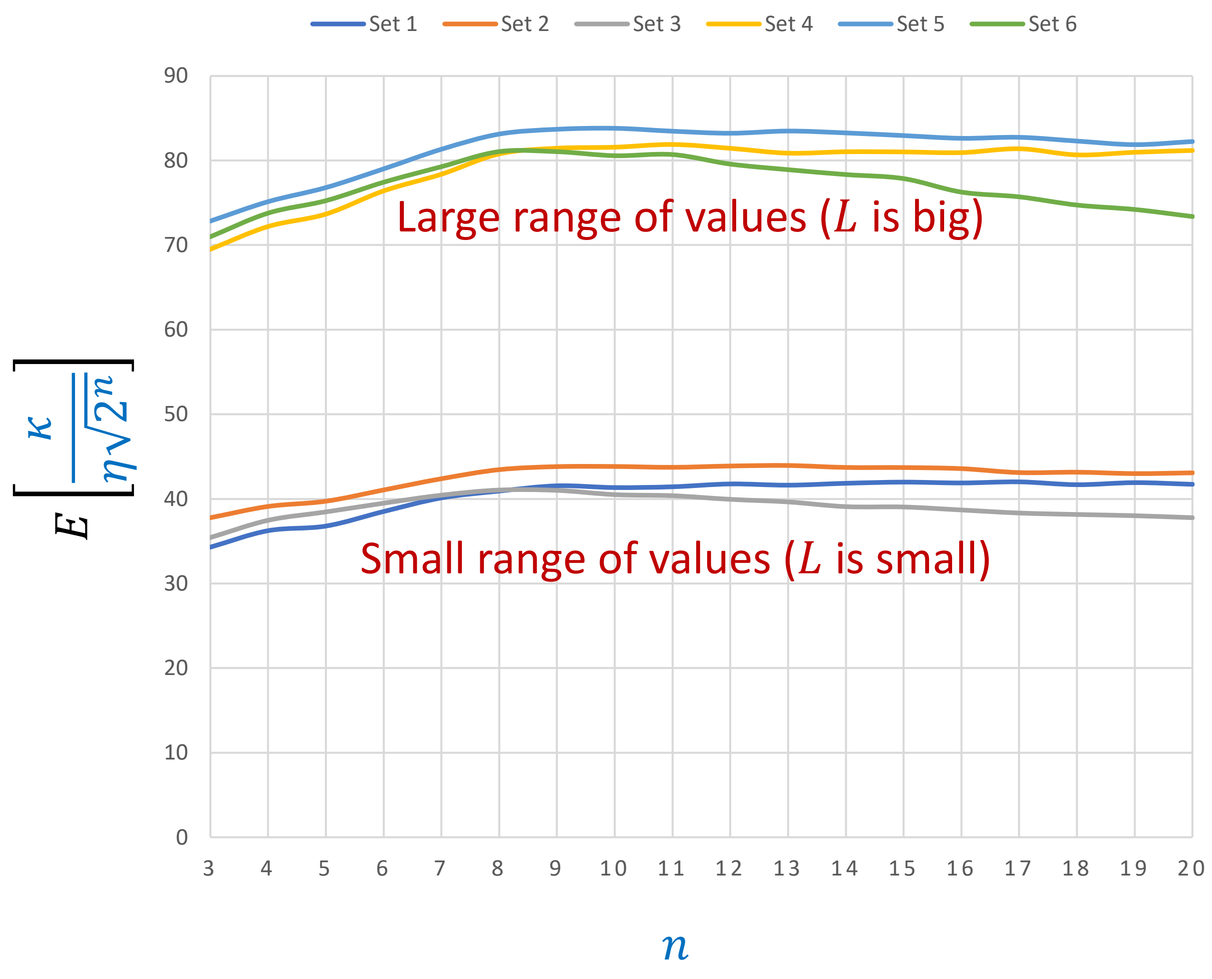
Grover's algorithm

- Grover's algorithm maximizes the probability to measure a solution $\boldsymbol{x}$ that has $f(\boldsymbol{x}) = 1$ using roughly $\sqrt{2^n/m}$ iterations, where m is the number of solutions for which $f(\boldsymbol{x}) = 1$.
- In our example, there is only one solution (i.e., 101) that has $f(\boldsymbol{x}) = 1$; that has $V \geq V^*$(i.e., $m = 1$).
- If $m = 1$, to find 101, Grover's algorithm needs roughly $\sqrt{2^n}$ iterations (and hence calls to $f(\boldsymbol{x})$).
- To find 101 on a classical computer, we need up to 2^ncalls to $f(\boldsymbol{x})$ if we use full enumeration ➔ Grover's algorithm achieves a quadratic speedup?
- When using Grover's algorithm to solve discrete optimization problems, we face two problems:
  - We don't know m.
  - We don't know V^*.

Binary Search Procedure (BSP)

- To solve these problems, we propose a Binary Search Procedure (BSP).
- First, to find the optimal value V^*, BSP initializes a minimum value V_{min} and a maximum value V_{max}. Next, binary search is used to evaluate different values of V until V^* is identified.
- For each value V, BSP also evaluates different values of m:
  - If, for a given value of m, a valid solution $\boldsymbol{x}$ is measured (that has value $V_{\boldsymbol{x}} \geq V$), we let $V_{min} = V + 1$.
  - If no valid solution can be found, we let $V_{max} = V - 1$.
- Million-dollar question: do we still achieve a quadratic speedup?

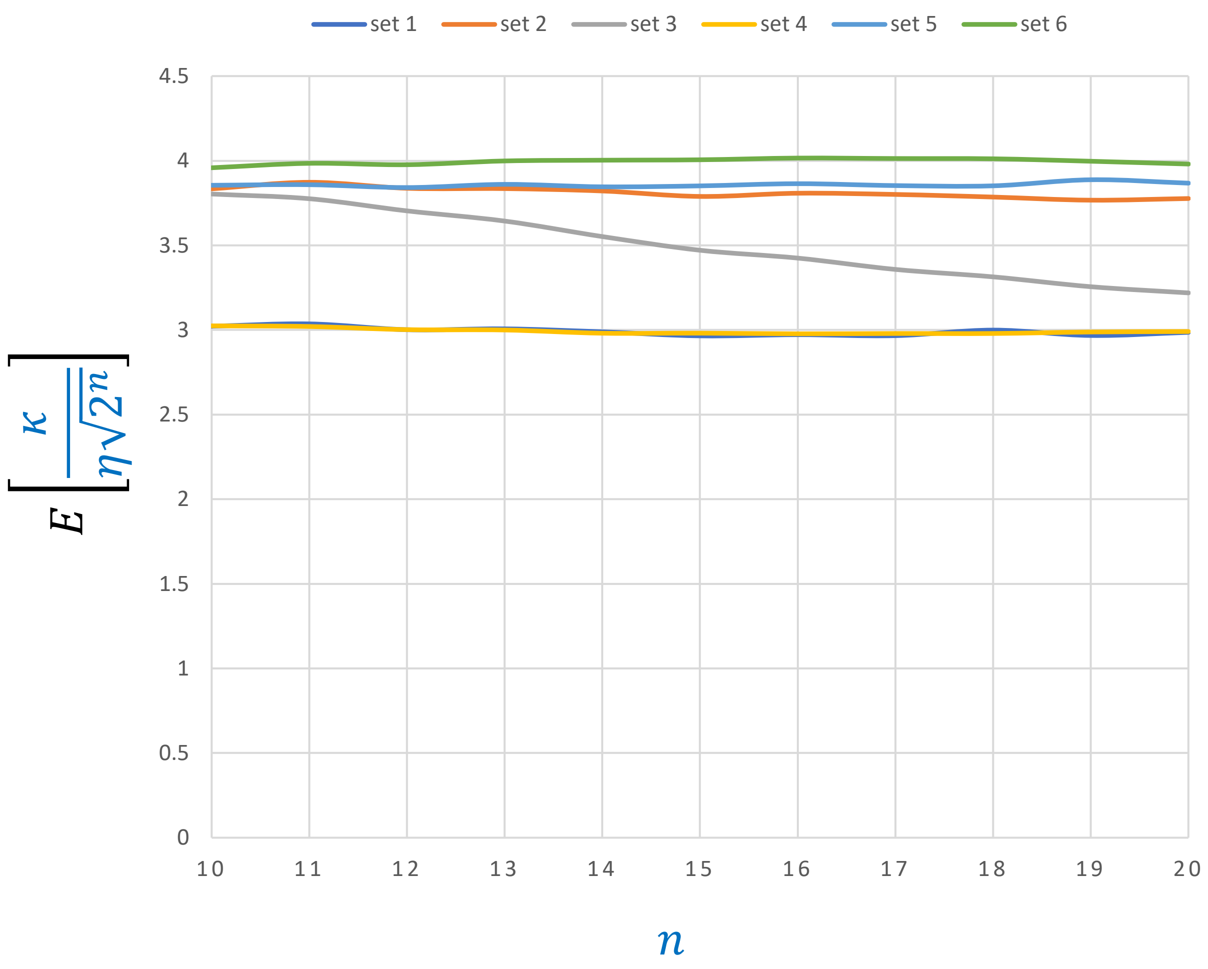
BSP: Results and complexity



- We use BSP to solve 1000 knapsack problems for:
  - Values of $n \in [3, \ldots, 20]$.
  - 6 problem sets
- We report the expected number of operations required to solve a knapsack problem (κ) divided by $\eta\sqrt{2^n}$.
- Complexity BSP is $O(\eta L\sqrt{2^n})$, where L is a logarithmic term depending on the range of values of knapsack items.
- No quadratic speedup due to logarithmic term L, however: can we do better?

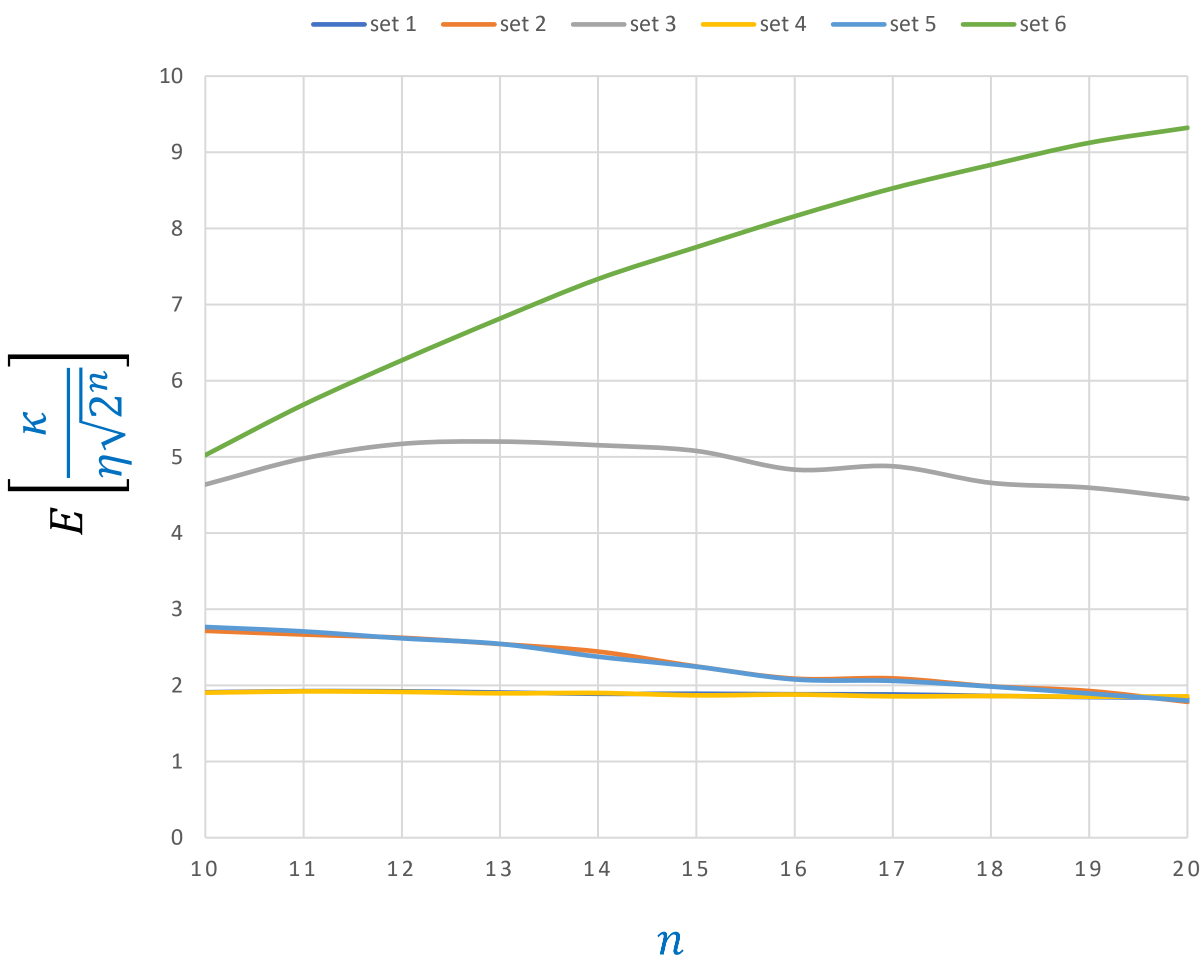

YES WE CAN!

Random Ascent Procedure (RAP)



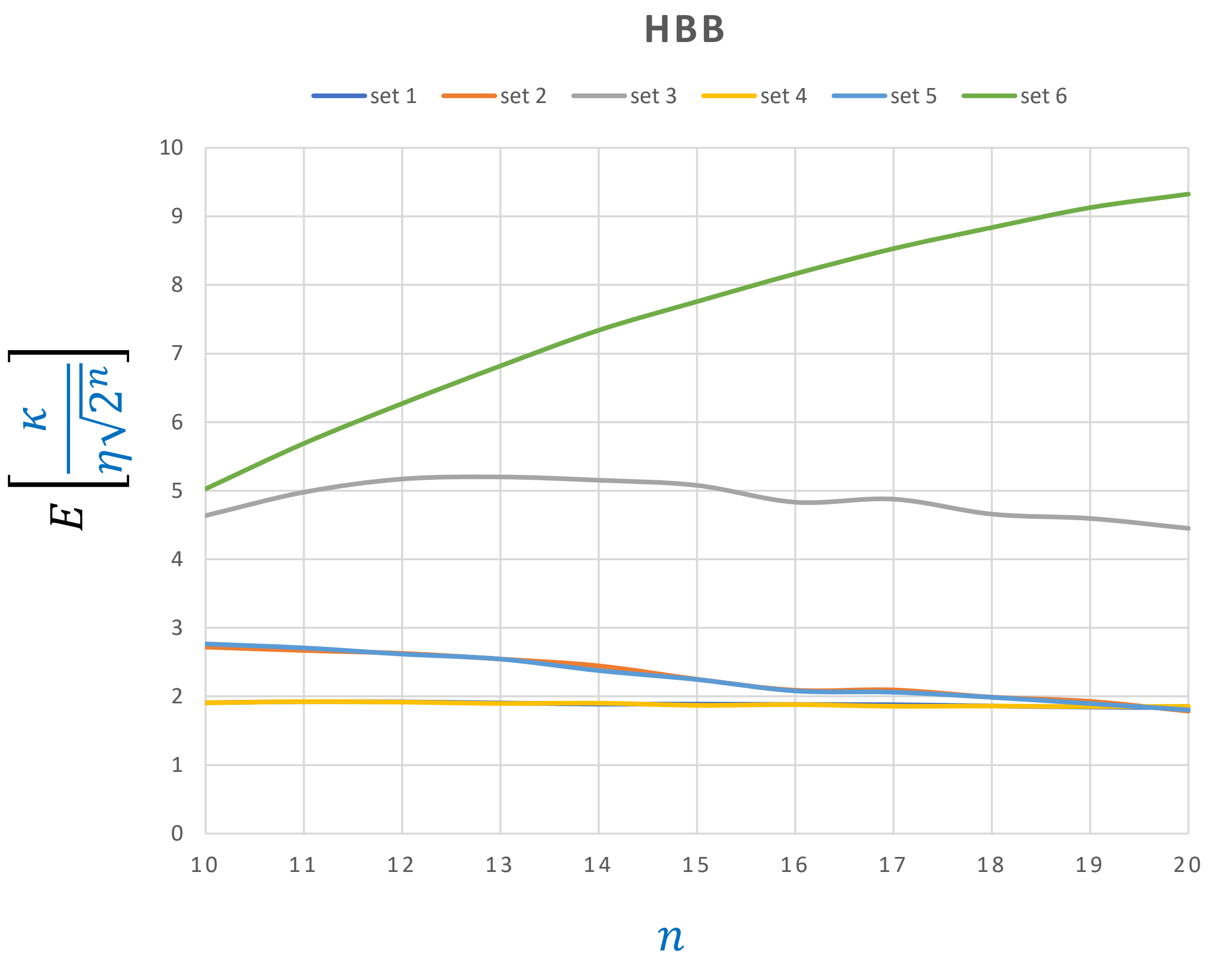
- Iterative procedure that uses Grover's algorithm to find a solution that has a better value than the best-found solution.
- If we measure, a better solution is chosen at random from the set of solutions that can still improve the best-found solution.
- RAP has worst-case expected complexity $O(\eta\sqrt{2^n})$.
- recall that for knapsack the best classical algorithm also has complexity $O(\eta\sqrt{2^n})$.

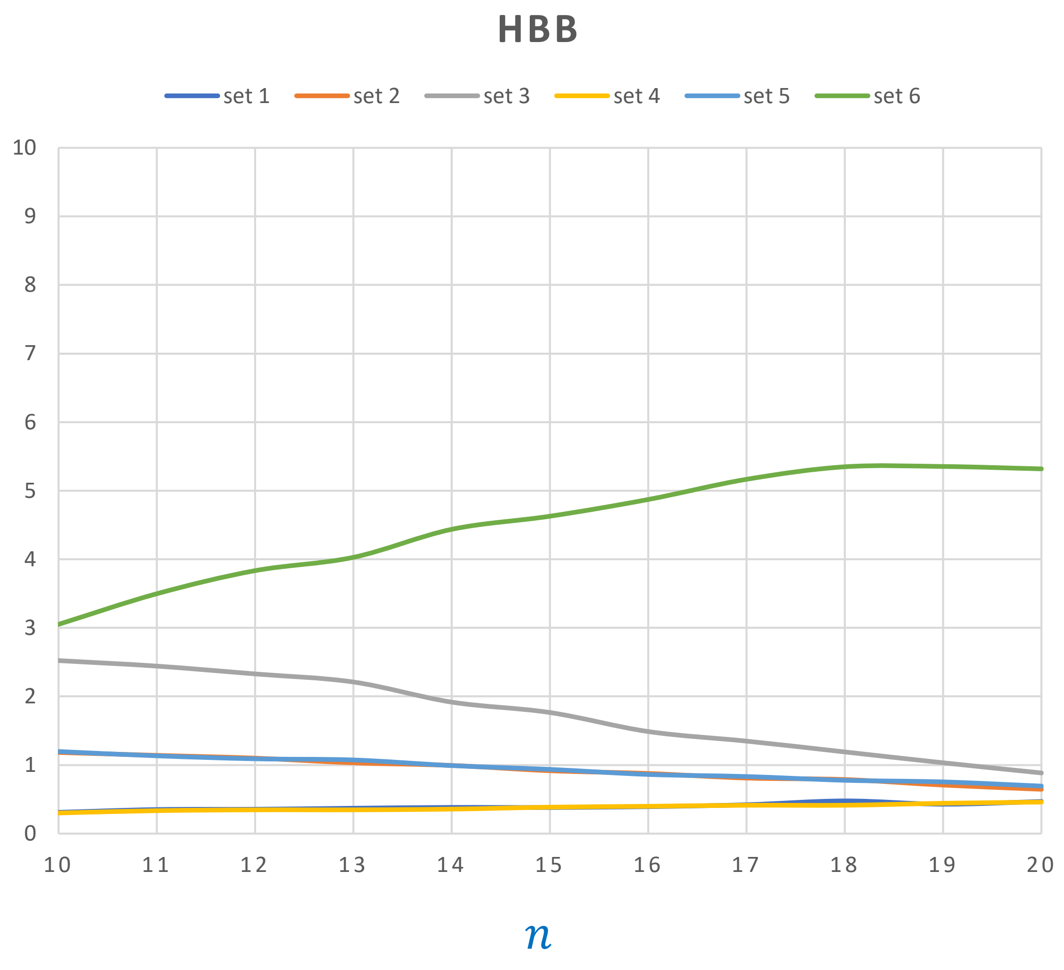
Hybrid Branch-and-Bound (HBB)



- Uses a tree that has n levels.
- At each level i, you create a node for each discrete value that can be assigned to decision variable x_i (i.e., you create a partial solution where the first i decision variables have been assigned a value).
- In each node, we use Grover's algorithm to see if we can find a solution for the remaining $n - i$ decision variables that improves the best-found solution:
  - If such a solution can be found, we branch.
  - If no solution can be found, we fathom the node.
- HBB also has complexity $O(\eta\sqrt{2^n})$.

HBB: Time to find optimal solution versus time to find optimal solution for 1st time





Conclusions

- We identified the problems faced when using Grover's algorithm to solve discrete optimization problems.
- We use Grover's algorithm as a subroutine in:
  - BSP (Binary Search Procedure).
  - RAP (Random Ascent Procedure).
  - HBB (Hybrid Branch-and-Bound).
- We use these algorithms to solve 108000 binary knapsack problems.
- We show that:
  - RAP & HBB require at most $O(\eta\sqrt{2^n})$ operations to find the optimal solution.
  - RAP & HBB match performance of best classical algorithms when solving knapsack.
  - RAP & HBB can also be used as heuristics using far less operations.
  - RAP & HBB can be used to solve **ANY** discrete optimization problem to optimality.

Want to know more?

- Read our three papers (currently under review):
  - Discrete optimization: A quantum revolution (Part I).
  - Discrete optimization: A quantum revolution (Part II).
  - Discrete optimization: Limitations of existing quantum algorithms.
- Available on SSRN and on my personal website (www.cromso.com).
- Contact us:

  - sc@cromso.com
  - l.fernando@ieseg.fr

EURO 2024 Copenhagen: Session on quantum computing

Invitation code:
7586e1c4

Stream:
Quantum Computing
Optimization

Session:
Quantum Computing &
Optimization III