Discrete optimization: A quantum revolution?

Stefan Creemers
Luis Fernando Pérez
October 18, 2023
Quantum Computing

Universal quantum computer

Quantum simulation

Grover-based algorithms

Quantum machine learning

Quantum factorization

Discrete optimization problems

Quantum annealing
Discrete optimization problems

• In the most general form:
  optimize $g(x_1, x_2, ..., x_n)$
  subject to
  $x_i \in \Omega_i, \forall i: 0 \leq i \leq n$
  (any other constraint)

• Where:
  • $g(x)$ is the objective function that evaluates assignment $x = \{x_1, x_2, ..., x_n\}$.
  • $n$ is the number of decision variables.
  • $x_i$ is the $i^{th}$ decision variable.
  • $\Omega_i$ 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 INFORMS
Basic unit of information: Classic vs quantum

**Classical computing**
- Bit.
- Can take on values 0 and 1.

![Classical computing](image)

**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 $x = \{x_1, x_2, \ldots, x_n\}$.
- Weight of $x$ is $W_x$.
- Value of $x$ is $V_x$.
- Function $f(x)$ evaluates whether solution $x$ is valid; has weight $W_x$ that does not exceed weight capacity $W$, and value $V_x$ is at least equal to $V^*$.

<table>
<thead>
<tr>
<th></th>
<th>$w_i$</th>
<th>$v_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

- $n = 3$
- $W = 4$
- $V^* = 5$

$x = \{x_1, x_2, x_3\}$

$W_x = \sum w_i x_i$

$V_x = \sum v_i x_i$

$f(x) = 1$ if $W_x \leq W$ and $V_x \geq V^*$
Solving the binary knapsack problem

• Classical computing:
  • Full enumeration requires $2^n = 8$ calls to function $f(x)$.
  • Each call to $f(x)$ requires $\eta$ operations.
  • In case of knapsack, $\eta = O(n) \Rightarrow$ full enumeration has complexity $O(n2^n)$.
  • Best classical algorithm to solve binary knapsack has complexity $O(n^2)$.

• Quantum computing:
  • Given a (uniform) superposition of three qubits, only a single call to $f(x)$ is required to obtain $f(x)$ for each possible solution $\Rightarrow$ complexity $O(n)$?
  • Each solution, however, has probability $2^{-n} = 0.125$ to be measured $\Rightarrow$ we only have a 12.5% chance to measure 101.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$w_i$</th>
<th>$v_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

$n = 3$ $W = 4$ $V^* = 5$

$x = \{x_1, x_2, x_3\}$ $W_x = \sum w_i x_i$ $V_x = \sum v_i x_i$

$f(x) = 1$ if $W_x \leq W$ and $V_x \geq V^*$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$W_x$</th>
<th>$V_x$</th>
<th>$f(x)$</th>
<th>$P(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.125</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0.125</td>
</tr>
<tr>
<td>010</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0.125</td>
</tr>
<tr>
<td>110</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0.125</td>
</tr>
<tr>
<td>001</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0.125</td>
</tr>
<tr>
<td>101</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>0.125</td>
</tr>
<tr>
<td>011</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0.125</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
<td>6</td>
<td>0</td>
<td>0.125</td>
</tr>
</tbody>
</table>
Grover’s algorithm

• Grover’s algorithm maximizes the probability to measure a solution $x$ that has $f(x) = 1$ using roughly $\sqrt{2^n/m}$ iterations, where $m$ is the number of solutions for which $f(x) = 1$.

• In our example, there is only one solution (i.e., 101) that has $f(x) = 1$; 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 $f(x)$).

• To find 101 on a classical computer, we need up to $2^n$ calls to $f(x)$ if we use full enumeration $\Rightarrow$ 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 $V_{\text{min}}$ and a maximum value $V_{\text{max}}$. 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 $x$ is measured (that has value $V_x \geq V$), we let $V_{\text{min}} = V + 1$.
  • If no valid solution can be found, we let $V_{\text{max}} = V - 1$.

• Million-dollar question: do we still achieve a quadratic speedup?
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 ($\kappa$) divided by $\eta_2 n$.

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?
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 $O(\eta \sqrt{2^n})$. 

![Graph showing the performance of RAP with different sets of data.](image)
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 \( O(\eta \sqrt{2^n}) \).
RAP versus HBB (solving to optimality)
RAP vs HBB (finding optimal solution for 1\textsuperscript{st} time)

**RAP**

- Set 1
- Set 2
- Set 3
- Set 4
- Set 5
- Set 6

**HBB**

- Set 1
- Set 2
- Set 3
- Set 4
- Set 5
- Set 6

$E \left[ \frac{\kappa}{n \sqrt{27}} \right]$
RAP: Time to find optimal solution versus time to find optimal solution for 1st time
HBB: Time to find optimal solution versus time to find optimal solution for 1\textsuperscript{st} time

\[ E \left[ \frac{\kappa}{n^2 \sqrt{n}} \right] \]
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.
Will quantum computing cause a revolution in the field of discrete optimization?

Yes: 100%
No: 0%
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).

• Coming soon to arXiv.

• Contact us:
  • sc@cromso.com
  • l.fernando@ieseg.fr