WIKIPEDIA
The Free Encyclopedia

Quadratic unconstrained binary optimization

Quadratic unconstrained binary optimization (**QUBO**), also known as **unconstrained binary quadratic programming** (**UBQP**), is a combinatorial optimization problem with a wide range of applications from finance and economics to machine learning.[1] QUBO is an NP hard problem, and for many classical problems from theoretical computer science, like maximum cut, graph coloring and the partition problem, embeddings into QUBO have been formulated.[2][3] Embeddings for machine learning models include support-vector machines, clustering and probabilistic graphical models.[4] Moreover, due to its close connection to Ising models, QUBO constitutes a central problem class for adiabatic quantum computation, where it is solved through a physical process called quantum annealing.[5]

Definition

The set of binary vectors of a fixed length $n > 0$ is denoted by $\mathbb{B}^n$, where $\mathbb{B} = \{0, 1\}$ is the set of binary values (or *bits*). We are given a real-valued upper triangular matrix $Q \in \mathbb{R}^{n \times n}$, whose entries Q_{ij} define a weight for each pair of indices $i, j \in \{1, \ldots, n\}$ within the binary vector. We can define a function $f_Q : \mathbb{B}^n \to \mathbb{R}$ that assigns a value to each binary vector through

$$f_Q(x) = x^\top Q x = \sum_{i=1}^{n} \sum_{j=i}^{n} Q_{ij} x_i x_j$$

Intuitively, the weight Q_{ij} is added if both x_i and x_j have value 1. When $i = j$, the values Q_{ii} are added if $x_i = 1$, as $x_i x_i = x_i$ for all $x_i \in \mathbb{B}$.

The QUBO problem consists of finding a binary vector x^* that is minimal with respect to f_Q, namely

$$x^* = \arg\min_{x \in \mathbb{B}^n} f_Q(x)$$

In general, x^* is not unique, meaning there may be a set of minimizing vectors with equal value w.r.t. f_Q. The complexity of QUBO arises from the number of candidate binary vectors to be evaluated, as $|\mathbb{B}^n| = 2^n$ grows exponentially in n.

Sometimes, QUBO is defined as the problem of *maximizing* f_Q, which is equivalent to minimizing $f_{-Q} = -f_Q$.

Properties

- Multiplying the coefficients Q_{ij} with a positive factor $\alpha > 0$ scales the output of f accordingly, leaving the optimum x^* unchanged:

$$f_{\alpha Q}(x) = \sum_{i \le j} (\alpha Q_{ij}) x_i x_j = \alpha \sum_{i \le j} Q_{ij} x_i x_j = \alpha f_Q(x)$$

- Flipping the sign of all coefficients flips the sign of f's output, making x^* the binary vector that *maximizes* f_{-Q}:

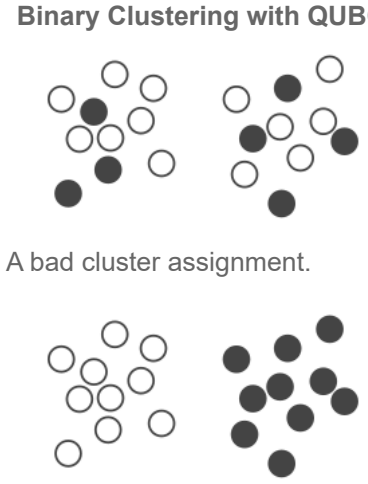
$$f_{-Q}(x) = \sum_{i \le j} (-Q_{ij}) x_i x_j = -\sum_{i \le j} Q_{ij} x_i x_j = -f_Q(x)$$

- If all coefficients are positive, the optimum is trivially $x^* = (0, \ldots, 0)$. Similarly, if all coefficients are negative, the optimum is $x^* = (1, \ldots, 1)$.
- If $\forall i \neq j : \ Q_{ij} = 0$, i.e., the bits can be optimized independently, then the corresponding QUBO problem is solvable in $\mathcal{O}(n)$, the optimal variable assignments x_i^* simply being 1 if $Q_{ii} < 0$ and 0 otherwise.

Applications

QUBO is a structurally simple, yet computationally hard optimization problem. It can be used to encode a wide range of optimization problems from various scientific areas.[6]

Cluster Analysis



Visual representation of a clustering problem with 20 points: Circles of the same color belong to the same cluster. Each circle can be understood as a binary variable in the corresponding QUBO problem.

As an illustrative example of how QUBO can be used to encode an optimization problem, we consider the problem of cluster analysis. Here, we are given a set of 20 points in 2D space, described by a matrix $D \in \mathbb{R}^{20 \times 2}$, where each row contains two cartesian coordinates. We want to assign each point to one of two classes or *clusters*, such that points in the same cluster are similar to each other. For two clusters, we can assign a binary variable $x_i \in \mathbb{B}$ to the point corresponding to the i-th row in D, indicating whether it belongs to the first ($x_i = 0$) or second cluster ($x_i = 1$). Consequently, we have 20 binary variables, which form a binary vector $x \in \mathbb{B}^{20}$ that corresponds to a cluster assignment of all points (see figure).

One way to derive a clustering is to consider the pairwise distances between points. Given a cluster assignment x, the values $x_i x_j$ or $(1 - x_i)(1 - x_j)$ evaluate to 1 if points i and j are in the same cluster. Similarly, $x_i(1 - x_j)$ or $(1 - x_i)x_j$ indicate that they are in different clusters. Let $d_{ij} \geq 0$ denote the Euclidean distance between points i and j. In order to define a cost function to minimize, when points i and j are in the same cluster we add their positive distance d_{ij}, and

subtract it when they are in different clusters. This way, an optimal solution tends to place points which are far apart into different clusters, and points that are close into the same cluster. The cost function thus comes down to

$$\begin{aligned} f(x) &= \sum_{i<j} d_{ij}(x_i x_j + (1-x_i)(1-x_j)) - d_{ij}(x_i(1-x_j) + (1-x_i)x_j) \\ &= \sum_{i<j} \left[4d_{ij}x_i x_j - 2d_{ij}x_i - 2d_{ij}x_j + d_{ij}\right] \end{aligned}$$

From the second line, the QUBO parameters can be easily found by re-arranging to be:

$$Q_{ij} = \begin{cases} 4d_{ij} & \text{if } i \neq j \\ -\left(\sum_{k=1}^{i-1} d_{ki} + \sum_{\ell=i+1}^{n} d_{i\ell}\right) & \text{if } i = j \end{cases}$$

Using these parameters, the optimal QUBO solution will correspond to an optimal cluster w.r.t. above cost function.

Connection to Ising models

QUBO is very closely related and computationally equivalent to the Ising model, whose Hamiltonian function is defined as

$$H(\sigma) = -\sum_{\langle i\ j\rangle} J_{ij}\sigma_i\sigma_j - \mu\sum_j h_j\sigma_j$$

with real-valued parameters h_j, J_{ij}, μ for all i, j. The *spin variables* σ_j are binary with values from $\{-1, +1\}$ instead of $\mathbb{B}$. Moreover, in the Ising model the variables are typically arranged in a lattice where only neighboring pairs of variables $\langle i\ j\rangle$ can have non-zero coefficients. Applying the identity $\sigma \mapsto 2x - 1$ yields an equivalent QUBO problem:[2]

$$\begin{aligned} f(x) &= \sum_{\langle i\ j\rangle} -J_{ij}(2x_i - 1)(2x_j - 1) + \sum_j \mu h_j(2x_j - 1) \\ &= \sum_{\langle i\ j\rangle} -4J_{ij}x_i x_j + 2J_{ij}x_i + 2J_{ij}x_j - J_{ij} + \sum_j 2\mu h_j x_j - \mu h_j \qquad \text{using } x_j = x_j x_j \\ &= \sum_{i=1}^{n}\sum_{j=1}^{i} Q_{ij}x_i x_j + C \end{aligned}$$

where

$$\begin{aligned} Q_{ij} &= \begin{cases} -4J_{ij} & \text{if } i \neq j \\ \sum_{\langle k\ i\rangle} 2J_{ki} + \sum_{\langle i\ \ell\rangle} 2J_{i\ell} + 2\mu h_i & \text{if } i = j \end{cases} \\ C &= -\sum_{\langle i\ j\rangle} J_{ij} - \sum_j \mu h_j \end{aligned}$$

As the constant C does not change the position of the optimum x^*, it can be neglected during optimization and is only important for recovering the original Hamiltonian function value.

References

1. Kochenberger, Gary; Hao, Jin-Kao (2014). "The unconstrained binary quadratic programming problem: a survey" (http://leeds-faculty.colorado.edu/glover/454%20-%20xQx%20survey%20article%20as%20published%202014.pdf) (PDF). *Journal of Combinatorial Optimization*. **28**: 58–81. doi:10.1007/s10878-014-9734-0 (https://doi.org/10.1007%2Fs10878-014-9734-0). S2CID 16808394 (https://api.semanticscholar.org/CorpusID:16808394).
2. Glover, Fred; Kochenberger, Gary (2019). "A Tutorial on Formulating and Using QUBO Models". arXiv:1811.11538 (https://arxiv.org/abs/1811.11538) [cs.DS (https://arxiv.org/archive/cs.DS)].
3. Lucas, Andrew (2014). "Ising formulations of many NP problems" (https://doi.org/10.3389%2Ffphy.2014.00005). *Frontiers in Physics*. **2**: 5. arXiv:1302.5843 (https://arxiv.org/abs/1302.5843). Bibcode:2014FrP.....2....5L (https://ui.adsabs.harvard.edu/abs/2014FrP.....2....5L). doi:10.3389/fphy.2014.00005 (https://doi.org/10.3389%2Ffphy.2014.00005).
4. Mücke, Sascha; Piatkowski, Nico; Morik, Katharina (2019). "Learning Bit by Bit: Extracting the Essence of Machine Learning" (https://web.archive.org/web/20200227143739/https://pdfs.semanticscholar.org/f484/b4a789e1563b91a416a7cfabbf72f0aa3b2a.pdf) (PDF). *LWDA*. S2CID 202760166 (https://api.semanticscholar.org/CorpusID:202760166). Archived from the original (https://pdfs.semanticscholar.org/f484/b4a789e1563b91a416a7cfabbf72f0aa3b2a.pdf) (PDF) on 2020-02-27.
5. Tom Simonite (8 May 2013). "D-Wave's Quantum Computer Goes to the Races, Wins" (http://www.technologyreview.com/view/514686/d-waves-quantum-computer-goes-to-the-races-wins/). MIT Technology Review. Retrieved 12 May 2013.
6. Ratke, Daniel (2021-06-10). "List of QUBO formulations" (https://blog.xa0.de/post/List-of-QUBO-formulations/). Retrieved 2022-12-16.

External links

- QUBO Benchmark (http://plato.asu.edu/ftp/qubo.html) (Benchmark of software packages for the exact solution of QUBOs; part of the well-known Mittelmann benchmark collection)
- Endre Boros, Peter L Hammer & Gabriel Tavares (April 2007). "Local search heuristics for Quadratic Unconstrained Binary Optimization (QUBO)" (http://portal.acm.org/citation.cfm?id=1231283). *Journal of Heuristics*. Association for Computing Machinery. **13** (2): 99–132. doi:10.1007/s10732-007-9009-3 (https://doi.org/10.1007%2Fs10732-007-9009-3). S2CID 32887708 (https://api.semanticscholar.org/CorpusID:32887708). Retrieved 12 May 2013.
- Di Wang & Robert Kleinberg (November 2009). "Analyzing quadratic unconstrained binary optimization problems via multicommodity flows" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2808708). *Discrete Applied Mathematics*. Elsevier. **157** (18): 3746–3753. doi:10.1016/j.dam.2009.07.009 (https://doi.org/10.1016%2Fj.dam.2009.07.009). PMC 2808708 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2808708). PMID 20161596 (https://pubmed.ncbi.nlm.nih.gov/20161596).