Literature Review

Raufs Dunamalijevs

Table of Contents

[1 Annealing in Combinatorial Optimization 2](#_Toc87859486)

[1.1 Annealing 2](#_Toc87859487)

[1.2 Simulated Annealing 2](#_Toc87859488)

[1.3 Quantum Annealing 3](#_Toc87859489)

[1.4 Digital Annealer 4](#_Toc87859490)

[2 QUBO 6](#_Toc87859491)

[2.1 QUBO model 6](#_Toc87859492)

[2.2 Constraints and Penalties 7](#_Toc87859493)

[2.3 Natural QUBO Formulation 8](#_Toc87859494)

[2.3.1 Minimum Vertex Cover 8](#_Toc87859495)

[2.3.2 Example Solution 8](#_Toc87859496)

[2.4 Non-Natural QUBO Formulation and Slack Variables 9](#_Toc87859497)

[2.5 Algorithms for solving QUBOs 10](#_Toc87859498)

[2.6 Relevant libraries 10](#_Toc87859499)

[2.6.1 PyQUBO 10](#_Toc87859500)

[2.6.2 qbsolv 10](#_Toc87859501)

[3 Penalty Optimisation Techniques 11](#_Toc87859502)

[3.1 Analytical 11](#_Toc87859503)

[3.2 Numerical 11](#_Toc87859504)

[3.3 Machine Learning 11](#_Toc87859505)

[4 Summary 12](#_Toc87859506)

[References 13](#_Toc87859507)

# 1 Annealing in Combinatorial Optimization

In this section we will discuss a class of metaheuristics that are inspired by the physical process of *annealing*. At the beginning, we will explain what it is. Then we will talk about multiple annealing metaheuristics that are relevant to this work. Finally, we will look at hardware named Digital Annealer.

## 1.1 Annealing

Annealing is a heat treatment process of matter like glass or metal (Collins, 1921; Wu and Fan, 2020). It involves heating up the material and then carefully cooling it down at a controlled rate. By following that process, it is possible to achieve a stable crystal structure.

If, however, this rate is exceeded, the resulting crystal will have defects as only locally optimal structures were allowed to form (Kirkpatrick, Gelatt and Vecchi, 1983). The lack of equilibrium will make such crystal metastable.

## 1.2 Simulated Annealing

Simulated annealing (SA) is a metaheuristic that uses the concept of temperature to approximate global optimum of a combinatorial optimization problem (Kirkpatrick, Gelatt and Vecchi, 1983; Bertsimas and Tsitsiklis, 1993) The temperature (*T*) is a variable that controls the likelihood of making locally non-optimal uphill moves. Just as in physical annealing, we start the optimization with a high temperature and decrease it at a controlled rate. If that rate is low enough, we are guaranteed to find a global optimum (Liang, Cheng and Lin, 2014). However, it is possible to find good approximations of the optima significantly faster at higher cooling rates.

The temperature concept allows the algorithm to explore the search space and find the optimal solution (Poole and Mackworth, 2017). The SA starts with a high temperature. At that point the algorithm acts like random walk, choosing random moves with high probability, which prevents us from being stuck at local optima – this is when exploration happens. At the end of the SA, when the temperature is low, the algorithm prioritizes advantageous moves. It acts like greedy algorithm and tries to find the best solution. As we shift from higher to lower temperatures, we gradually shift from exploring the search space to find the optimum.

Let *c* be current node, *n* be next node (neighbour of node *c*) and *evaluation(x)* be the function that evaluates the energy of the node *x*. Then the energy transition (*ΔE*) can be defined by the following formula:

*ΔE = evaluation(n) - evaluation(c)*

*ΔE* and *T* are the variables that will affect the probability of choosing node *n*, where *ΔE* is an energy transition (negative when minimising) that the proposed node will introduce, and *T* is controlling how *ΔE* influences the probability of choosing this node (Johnson *et al.*, 1989).

The *acceptance probability function* is the function that maps current node, proposed node, and Temperature to a probability. There are many functions that suit this purpose, the original being an exponential function (Kirkpatrick, Gelatt and Vecchi, 1983).

When maximizing, the negative sign should be removed from the formula.

If that acceptance probability is higher than a uniformly generated random number, then node *c* is rejected in favour of node *n*. However, when (downhill movement), we will always choose the proposed node (Kirkpatrick, Gelatt and Vecchi, 1983).

Pseudocode of SA (Johnson *et al.*, 1989) is given by:

c ← initial node

**while** T > threshold

n ← generate next node(c)

ΔE ← evaluate(n) – evaluate(c)

**if** ΔE ≤ 0 t

c ← n

**else if** P(c, n, T) ≥ random(0, 1)

c ← n

**end**

T ← cooling(T)

**end**

## 1.3 Quantum Annealing

Quantum annealing (QA) is a metaheuristic that uses quantum-mechanical fluctuations to find a global minimum of a function (Ohzeki and Nishimori, 2010; Tanaka and Tamura, 2012). In QA we start with a strong quantum field and gradually decrease its strength (quantum fluctuation) to find a ground state that is optimal. This is analogous to SA, where we decreased the temperature (thermal fluctuation) to achieve the same result.

QA begins in a uniform superposition of all possible states, but as we decrease the strength of quantum field, we approach a single solution state (Farhi *et al.*, 2001).

In SA neighbouring nodes are located 1 move away for the current node. In QA the neighbouring nodes are in a certain range. This range is defined by the strength of the quantum field, slowly ceasing in the process of annealing. Transition from one state to another can happen within that range. If there is a short wall between current state, which happens to be a local minimum, and the global minimum, SA algorithm will need to make a series of disadvantageous moves to reach a better solution or in other words *“climb”* over the wall. This is avoided in QA as there is a capacity to tunnel through short walls if the quantum field is strong enough, making QA faster compared to SA when dealing with search spaces with many thin barriers (Crosson and Harrow, 2016). This effect is called *quantum tunnelling*. Comparison of thermal hopping and quantum tunnelling can be seen on Figure 1.

Diagram

Description automatically generated

Figure 1. Insert text

The fundamental feature and the main benefit of QA is that it rapidly samples a wide range of configurations when exploring the energy landscape (Higham and Bedford, 2021). Quantum Annealer has been developed and commercialised by D-wave (Johnson *et al.*, 2011) and is currently used to solve real-world problems (D-Wave Systems, 2021).

## 1.4 Digital Annealer

Digital Annealer (DA) is quantum-inspired custom CMOS hardware developed by Fujitsu Laboratories (Aramon *et al.*, 2019). Its algorithm is based on SA, but has 3 major differences:

1. Instead of considering a single neighbouring node and choosing it if the acceptance probability is higher than a uniform random number, DA considers all neighbouring nodes in parallel. If more than a single node was accepted, 1 of them will be chosen uniformly at random to move into. This is advantageous as in SA if the node was rejected, we need to generate and then consider a new one from the same position, which takes time. In DA, however, all the nodes are considered in parallel and the acceptance probability per “turn” is therefore higher.
2. DA uses *dynamic offset*, which means that if no nodes were accepted, the next acceptance probability will be artificially increased to help the algorithm overcome narrow barriers.
3. All runs begin with the same random state to save time (in SA that runs in parallel the initial states are generated individually for each of the runs). Otherwise, it would need to evaluate the new initial node and all the new neighbouring nodes for each run.

These differences can be observed in DA pseudocode (Aramon *et al.*, 2019):

initial state ← random state

**for** each run

current state ← initial state

Eoffset ← 0

**for** each step

update temperature if needed

**for** each neighbour *(in parallel)*

calculate ∆Eneighbour

consider neighbour using ∆Eneighbour - ∆Eoffset

if neighbour accepted, record

**end**

**if** accepted neighbours ≥ 1

choose one accepted neighbour uniformly at random

update current state and the energy landscape *(in parallel)*

Eoffset ← 0

**else**

increase Eoffset

**end**

**end**

**end**

Although it is hard to make comparisons as the technology is quickly developing, it has been shown that DA can be faster and produce more precise solutions compared to D-Wave 2000Q (penultimate generation of D-Wave’s QA devices) in certain scenarios (Ohzeki *et al.*, 2019). DA’s are used in industry and science (Fujitsu, no date; Snelling *et al.*, 2020)

Currently QA devices are expensive and difficult to run as the technology is still under development (Şeker *et al.*, 2020). D-Wave 2000Q also suffers from qubit noise that results in lower precision (Katzgraber and Novotny, 2018), and sparse connectivity between spins that makes it less effective (Hamerly *et al.*, 2018). These problems are actively worked on and some of them are addressed in more recent QA devices (Dattani, Szalay and Chancellor, 2019). As QA technology develops, it will likely outperform DA largely due to its quantum parallelism (Boyd, 2018).

# 2 QUBO

Quadratic Binary Unconstrained Optimization (QUBO) is a mathematical formulation that can be applied to many combinatorial optimization (CO) problems (Kochenberger *et al.*, 2014). Problems formulated in QUBO can be subsequently solved using QUBO solvers. QA devices developed by D-Wave can solve CO problems that are formulated as QUBO or Ising models (Cruz-Santos, Venegas-Andraca and Lanzagorta, 2019). It has been shown that QUBO and Ising models are equivalent (Lucas, 2014), therefore one can be derived from another. DA’s can only solve problems that are formulated in QUBO (Aramon *et al.*, 2019).

There are many combinatorial optimization problems from industry, government and science that can be reformulated in QUBO (Glover, Kochenberger and Du, 2019).

In this section we will discuss the structure of QUBO, basics of reformulating constrained CO problems into equivalent unconstrained QUBO models, algorithms that can be used to solve these problems and relevant programming libraries that can be useful when dealing with QUBO’s. We will introduce the concept of penalty, which will be explained in greater details in the next section.

## 2.1 QUBO model

QUBO models have been described in great details in the work of Glover, Kochenberger and Du (2019). This subsection will summarize the key information relevant to this project.

The QUBO model can be expressed in the following way:

where *y* is the value to be optimized for, *x* is a vector of decision variables and *Q* is a square matrix with coefficients. QUBO models are unconstrainted, the only restriction being that every variable in decision vector *x* should be either 0 or 1. It is self-contained as all the information needed for the optimisation is stored in the matrix Q. QUBO problems are NP-hard.

A simple example of how to convert a Boolean function into QUBO can be demonstrated with the following minimization problem:

Our function has 2 parts: linear and quadratic. As every *xj* belongs to {0, 1}, we can easily make the linear part quadratic (because 02=0 and 12=1), making the entire function quadratic.

We can then write it using matrixes in the form of , which is a QUBO model formula shown at the beginning.

## 2.2 Constraints and Penalties

In the previous subsection it has been shown how an unconstrained minimization problem can be reformulated as QUBO. Constrained problems can be reformulated in this model too. The constraints will be expressed as quadratic penalties, which will be added to the original objective function. The more constraints are broken, the larger is the overall penalty added. When none of the constraints are broken, the penalty imposed will be equal to 0.

Since we are trying to solve a minimization problem, solutions that brake constraints and impose penalty on the objective function will be avoided.

ФОРМУЛА: y = original function + M\*penalty function

М – это позитивный скаляр пенальти (Lewis). Он контралирует то, с какой эффект будет иметь нарушение правил на объективную функцию. Можно использовать несколько шкаляров для разных констрейнтов, если мы хотим, чтобы поломка разных констрейнтов имела разное влияние на объективную функцию (то есть имела разную важность). В некоторых проблемах констрейнты, или некоторые из них, не являются хард и их можно ломать. В таких случаях более маленькое значение М помогает придавать им меньшее значение. Но обычно используется один шкаляр на все (Гловер).

If the penalty coefficient is too low, the broken constraints will be undervalued, and the solution produced by the optimizer will be infeasible. On the other hand, if the penalty coefficient is too large, the solution process will be negatively impacted as the penalties will overwhelm the objective function making it harder to differentiate between good and bad solutions (гловер)

It is possible to use domain expertise to select an acceptable 𝑃 in some cases (Glover, Kochenberger and Du, 2019), but it has also been shown that some problems can have solutions of better quality if you choose penalty coefficient carefully (Şeker, Tanoumand and Bodur, 2020). Finding an optimal penalty value is not trivial and different techniques have been proposed to estimate it (Verma and Lewis, 2020; Huang et al., 2021).

Квадратные пенальти для некоторых видов констрейнтов уже известны. Некоторые из них показаны на таблице ниже.

Table

Description automatically generated

Рисунок 1. Все вериеблы в таблице бинарны

Представим, что в нашей проблеме либо x1 либо x2 должен быть равен 1. Но оба они равняться нулю или единице не могут. Этот констрейнт можно выразить следующей формулой:

Тогда используя пенальти из таблицы 1, мы можем выразить объективную функцию которую минимизируем подобным образом, где f(x) это функия без выраженного констрейнта

Можно увидеть, что пенальти не будет импозд только в том случае, если оба вэриэбла равны 1 или 0, то есть, когда ломается констрейнт.

Glover

Verma Lewis

## 2.3 Natural QUBO Formulation

### 2.3.1 Minimum Vertex Cover

Есть ряд проблем, которые натурально ложатся на кубо. Одним из таких является проблема минимум вертекс кавер. В ней нам дан undirected граф с вертексами и углами, репрезентируемыми сетами V и Е респективно. Вертекс кавер – это субсет из вертексов V, который каверит все эджи E. На фигуре 2 можно увидеть такой граф и сравнение обычных субсетов и вертекс каверов.

ФИГУРА 2

Вектор решений x может быть реализован таким образом. Если вертекс j находится в кавере, xj = 1. Если нет xj = 0. Тогда вот как выглядит наша минимизационная функция до того как мы законсидерили констрейнты.

ФОРМУЛА

Для решения этой проблемы нам нужно, чтобы каждый эдж был закаверин. Это так же можно выразить так: у каждого эджа должен быть хотя бы один вертекс, который принадлежит к каверу. Это констрейнты и они могут быть выражен в подобной форме:

ФОРМУЛА

Посмотрев на таюлицу 1, можно увидеть пенальти эквивалентное такому роду констрейнтов. Используя её, мы можем сделать следующую минимализационную функцию

ФОРМУЛА

Text

Description automatically generated with medium confidence

Данная формула может быть выражена как , где c – это константа, которая не имеет никакого влияния на процесс минимизации, поэтому может быть убрана. This leaves us with a QUBO model.

### 2.3.2 Example Solution

Нам дан следующий график, для которого надо найти MVC (Левис):

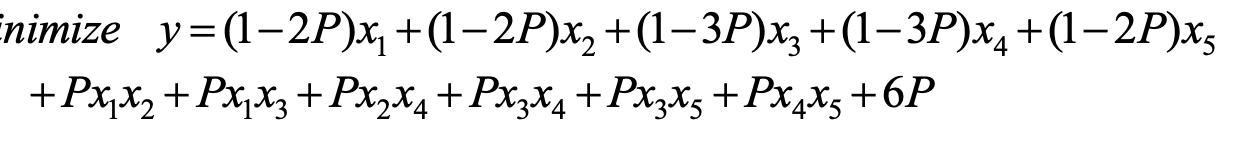
*A picture containing diagram

Description automatically generated*

По вышеописанной формуле, эту проблему можно выразить и приобразить вот так:

Text, letter

Description automatically generated



Если убрать консанту 6M в конце, которая не влияет на оптимизацию (если надо, мы её можем прибавить в самом конце, когда сама минимизация будет завершена), и присвоить М арбитратрное значение 8, мы получим следующую модель QUBO.

Q=[matrix]

Y = xtQx

Со следующим решением

Y = -45

X = 01101

V = 2, 3, 5

FIGURE SOLUTION

## 2.4 Non-Natural QUBO Formulation and Slack Variables

### 2.4.1 QUBO Formulation

Не все СО проблемы имеют натуральную формуляцию в кубо. Так же квадратные пенальти некоторых констрейнтов могут быть неизвестны. Их тоже можно реформулировать в QUBO, что и будет описано в этой субсекции.

Представим следующую оптимизационную проблему для минимизациии, где констрейнты пока есть, но не выражены:

Text, letter

Description automatically generated

Сделать норм математику

Учитывая, что констрейнты проблемы содержат только интеджеры, их всегда можно записать в подобной форме (пример этого будет показан в субсекции со слек вериеблами)

FORMULA

Ax=b

Где х принадлежит бинарным

Тогда квадратное пенальти, которое надо добавить на объективную, можно выразить таким образом

Quadratic penalty(x) = M(Ax-b)t(Ax-b)

А получившуюся объективную функцию с констрейнтами:

Text, letter

Description automatically generated

Как говорилось в прошлой секции, константу с на время оптимизации можно выкинуть, и если надо добавить в конце, когда мы нашли солюшн. Таким образом, мы получаем стандартную y=xtQx QUBO модель для минимизации из дженерал констрейнд проблемы.

### 2.4.2 Slack Variables

Представим, что у нас есть следующий inequality constraint

4x1 + 5x3 – x3 <= 6

ФОРМУЛА

Его можно сделать equality констрейнтом добавив слак вериебл, который не может превышать 7 и принижать -3, так как минимальное значение, которое функция может принять -1 (с конфигурацией 001), а максимальное 9 (с конфигурацией 110)

ФОРМУЛА

4x1 +5x2 – x3 + s = 6

-3 <= s <= 7

Если выразить s бинарной экспансией, которая тем или иным образом будет сатисфаить все комбинанции xj в нашей экволити, то мы получим следующее

ФОРМУЛА

4x1 + 5x2 – x3 + s1 + 2s2 + 4s3 = 6

А это уже можно записать в следующей форме, если выразить бинарные слеки тоже внутри вектора х.

FORMULA

Ax=b

Тогда, как и говорилось в субсекции посвящённой теории, этот констрейнт можно выразить как квадратное пенальти, ФОРМУЛА. Его в свою очередь можно потом использовать как показано в первой секции, чтобы добиться стандартной модели кубо.

## 2.5 Algorithms for solving QUBOs

## 2.6 Relevant libraries

### 2.6.1 PyQUBO

### 2.6.2 qbsolv

# 3 Penalty Optimisation Techniques

In this section we will present multiple approaches to penalty optimisation.

https://www.sciencedirect.com/science/article/abs/pii/S1572528620300281#b3

## 3.1 Analytical

## 3.2 Numerical

## 3.3 Machine Learning

# 4 Summary

# References

Aramon, M. *et al.* (2019) ‘Physics-Inspired Optimization for Quadratic Unconstrained Problems Using a Digital Annealer’, *Frontiers in Physics*, 0(APR), p. 48. doi: 10.3389/FPHY.2019.00048.

Bertsimas, D. and Tsitsiklis, J. (1993) ‘Simulated Annealing’, *Statistical Science*, 8(1), pp. 10–15.

Boyd, J. (2018) ‘Fujitsu’s CMOS Digital Annealer Produces Quantum Computer Speeds’, *IEEE Spectrum*, 28 May. Available at: https://spectrum.ieee.org/ieee-courses-on-digital-transformation (Accessed: 12 November 2021).

Collins, E. F. (1921) ‘ELECTRICALLY HEATED GLASS ANNEALING LEHR1’, *Journal of the American Ceramic Society*, 4(5), pp. 335–349. doi: 10.1111/J.1151-2916.1921.TB18664.X.

Crosson, E. and Harrow, A. W. (2016) ‘Simulated Quantum Annealing Can Be Exponentially Faster than Classical Simulated Annealing’.

Cruz-Santos, W., Venegas-Andraca, S. E. and Lanzagorta, M. (2019) ‘A QUBO Formulation of Minimum Multicut Problem Instances in Trees for D-Wave Quantum Annealers’, *Scientific Reports 2019 9:1*, 9(1), pp. 1–12. doi: 10.1038/s41598-019-53585-5.

D-Wave Systems (2021) *Volkswagen: Navigating Tough Automotive Tasks with Quantum Computing*. Available at: www.dwavesys.com/d-wave-launch (Accessed: 5 October 2021).

Dattani, N., Szalay, S. and Chancellor, N. (2019) ‘Pegasus: The second connectivity graph for large-scale quantum annealing hardware’.

Farhi, E. *et al.* (2001) ‘A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem’.

Fujitsu (no date) *Reducing traveling distance for parts picking operations by up to 45%*. Available at: https://www.fujitsu.com/global/services/business-services/digital-annealer/case-studies/201804-fjit.html (Accessed: 5 October 2021).

Glover, F., Kochenberger, G. and Du, Y. (2019) ‘Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models’, *4OR*, 17(4), pp. 335–371. doi: 10.1007/S10288-019-00424-Y.

Hamerly, R. *et al.* (2018) ‘Scaling advantages of all-to-all connectivity in physical annealers: the Coherent Ising Machine vs. D-Wave 2000Q’.

Higham, C. F. and Bedford, A. (2021) ‘Quantum Deep Learning: Sampling Neural Nets with a Quantum Annealer’. Available at: https://arxiv.org/abs/2107.08710v1 (Accessed: 11 November 2021).

Johnson, D. S. *et al.* (1989) ‘Operations Research Society of America This content downloaded from 128’, 37(6).

Johnson, M. W. *et al.* (2011) ‘Quantum annealing with manufactured spins’, *Nature 2011 473:7346*, 473(7346), pp. 194–198. doi: 10.1038/nature10012.

Katzgraber, H. G. and Novotny, M. A. (2018) ‘How Small-World Interactions Can Lead to Improved Quantum Annealer Designs’, *Physical Review Applied*, 10(5), p. 054004. doi: 10.1103/PHYSREVAPPLIED.10.054004/FIGURES/13/MEDIUM.

Kirkpatrick, S., Gelatt, ; C D and Vecchi, ; M P (1983) ‘Optimization by Simulated Annealing’, *New Series*, 220(4598), pp. 671–680.

Kochenberger, G. *et al.* (2014) ‘The unconstrained binary quadratic programming problem: A survey’, *Journal of Combinatorial Optimization*, 28(1), pp. 58–81. doi: 10.1007/S10878-014-9734-0/TABLES/1.

Liang, F., Cheng, Y. and Lin, G. (2014) ‘Simulated Stochastic Approximation Annealing for Global Optimization With a Square-Root Cooling Schedule’, *https://doi.org/10.1080/01621459.2013.872993*, 109(506), pp. 847–863. doi: 10.1080/01621459.2013.872993.

Lucas, A. (2014) ‘Ising formulations of many NP problems’, *Frontiers in Physics*, 0, p. 5. doi: 10.3389/FPHY.2014.00005.

Ohzeki, M. *et al.* (2019) ‘Control of Automated Guided Vehicles Without Collision by Quantum Annealer and Digital Devices’, *Frontiers in Computer Science*, 1, p. 9. doi: 10.3389/FCOMP.2019.00009/BIBTEX.

Ohzeki, M. and Nishimori, H. (2010) ‘Quantum annealing: An introduction and new developments’.

Poole, D. L. and Mackworth, A. K. (2017) ‘Artificial Intelligence: Foundations of Computational Agents’, pp. 136–137. doi: 10.1017/9781108164085.

Şeker, O. *et al.* (2020) ‘Digital Annealer for quadratic unconstrained binary optimization: a comparative performance analysis’. Available at: http://arxiv.org/abs/2012.12264 (Accessed: 3 October 2021).

Snelling, D. *et al.* (2020) ‘A Quantum-Inspired Approach to De-Novo Drug Design’. doi: 10.26434/CHEMRXIV.12229232.V1.

Tanaka, S. and Tamura, R. (2012) ‘QUANTUM ANNEALING AND QUANTUM FLUCTUATION EFFECT IN FRUSTRATED ISING SYSTEMS’.

Wu, H. and Fan, G. (2020) ‘An overview of tailoring strain delocalization for strength-ductility synergy’, *Progress in Materials Science*, 113, p. 100675. doi: 10.1016/J.PMATSCI.2020.100675.