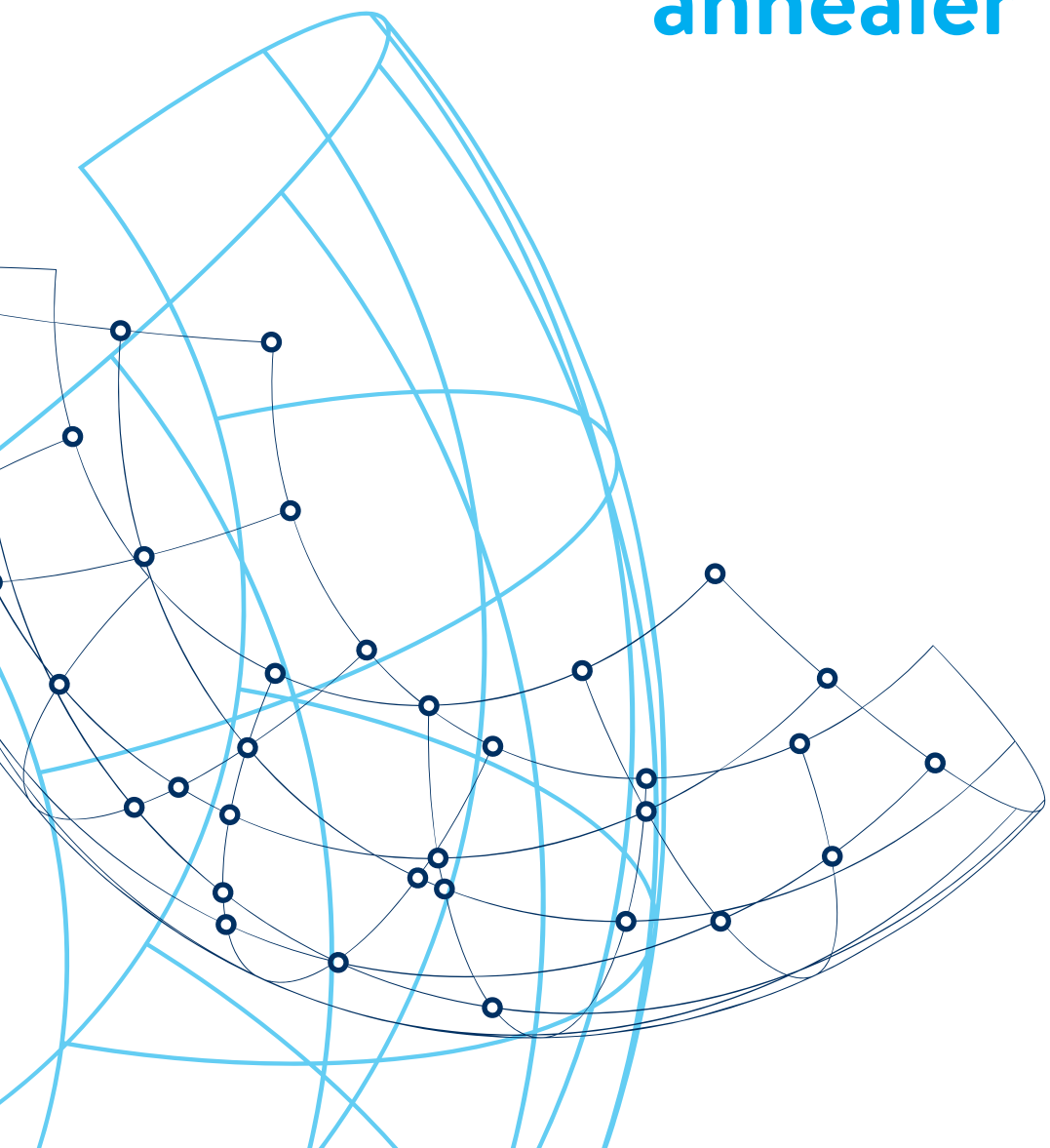
1QBit

Finding optimal arbitrage opportunities using a quantum annealer



White Paper

Finding optimal arbitrage opportunities using a quantum annealer

Gili Rosenberg

Abstract

We present two formulations for finding optimal arbitrage opportunities as a quadratic unconstrained binary optimization problem, which can be solved using a quantum annealer. The formulations are based on finding the most profitable cycle in a graph in which the nodes are the assets and the edge weights are the conversion rates. The edge-based formulation is simpler, whereas the node-based formulation allows for the identification of specific optimal arbitrage strategies, while possibly requiring fewer variables. In addition, an alternative form is presented which allows the arbitrage opportunities that best balance profit and risk to be found, based on the trader's risk aversion. We discuss considerations for usage in practice. In particular, we suggest an application to illiquid assets and present an illustrative example.

Keywords: arbitrage, quantum annealing, optimization

1 Introduction

Arbitrage encompasses a wide range of ways to make a profit from differing prices for the same asset in different markets. An example is cross-currency arbitrage—one might find that converting a sum of money from USD to CAD, then to EUR and then back to USD, results in a small profit after transaction costs.

The problem of arbitrage detection, that is, finding whether at least one arbitrage opportunity exists (and what it is) is a textbook problem [2]. It is commonly solved by defining a directed graph in which the nodes are the assets and the edge weights are equal to minus the logarithm of the conversion rate, and searching for negative cycles. This problem can be solved in polynomial time, for example, using Bellman-Ford's algorithm with time complexity $\mathcal{O}(|V||E|)$ (where $|V|$ is the number of vertices/assets and $|E|$ is the number of conversion rates), or Floyd-Warshall's algorithm with time complexity $\mathcal{O}(|V|^3)$, since they terminate if a negative cycle is found.

In contrast, the problem of finding the most profitable arbitrage opportunity (i.e., finding the negative cycle with the

most negative weight) of arbitrary length is NP-hard, and hence the time complexity of an algorithm solving it to optimality is expected to be exponential. In practice, traders tend to be more interested in patterns that can actually be traded, for example, liquid markets, simpler combinations, fewer orders, less leg risk, and so on.

However, the problem of finding the most profitable arbitrage opportunity over a large set of instruments is of more than mere theoretical interest. For example, commodities are often traded in strips of three months, six months, and twelve months. To include these in an arbitrage cycle requires extra edges and rules for their traversal. Moreover, since the existence of an arbitrage cycle indicates that not all market participants are acting on the same information, even a long and untradeable (from a practical standpoint) cycle can provide useful information, for instance, by showing deviations from market equilibrium.

In practice, the difference between arbitrage detection and finding the best arbitrage opportunity can be large. Consider an example in which two arbitrage opportunities exist, one with a tiny profit and the other with a huge profit. A trader would be interested in the larger profit, but the above approach (via the Bellman-Ford or Floyd-Warshall algorithms) stops when it finds the first arbitrage opportunity, which could often be the former. In addition, our approach (presented below) is able to find not only the best (most profitable) arbitrage opportunity but also others that are near-best. This is by virtue of the quantum annealer returning a sample containing near-optimal solutions.

2 Formulation (edge-based)

We construct a directed graph in which each node corresponds to an asset, and each directed edge is weighted with the corresponding conversion rate. A conversion rate c_{ij} means that if you have one unit of asset i, then you could convert it to c_{ij} units of asset j at no additional cost. In general, the conversion rate from asset i to asset j is different from the conversion rate from asset j to asset i (i.e., $c_{ij} \neq c_{ji}$, in general). The transaction costs are assumed to be included in the conversion rate, so there is a relative transaction cost (per unit) and no fixed transaction cost (per transaction). The optimization problem we wish to solve is to find the most profitable cycle in this graph.

To find the most profitable arbitrage opportunity, we seek to maximize the product of the conversion rates in a cycle. Let us define a binary decision variable x_{ij} such that it is equal to one if the respective edge is included in the chosen cycle, and zero otherwise, and denote the set of nodes (assets) by V, the number of nodes (assets) by $N = |V|$, and the set of edges by E. The profit of a given cycle can be written as a product over the conversion rates $\prod[(c_{ij} - 1)x_{ij} + 1] - 1$, which is a polynomial of a degree equal to the number of conversion rates (or edges). We notice that by maximizing the logarithm of the product instead, the order can be lowered to linear, $\sum x_{ij} \log c_{ij}$ (discarding the constant). The cycle constraint can be written by way of a conservation of flow constraint [3], with an additional constraint that forbids passing through a node twice, giving the optimization problem

$$\begin{aligned} \text{max} \quad & \sum_{(i,j)\in E} x_{ij} \log c_{ij} \\ \text{s.t.} \quad & \sum_{j,(i,j)\in E} x_{ij} = \sum_{j,(j,i)\in E} x_{ji} \quad \text{for all } i \in V, \\ & \sum_{j,(i,j)\in E} x_{ij} \leq 1 \quad \text{for all } i \in V. \end{aligned}$$

In order to use a quantum annealer to solve this problem, we convert it to a quadratic unconstrained binary optimization (QUBO) problem by rewriting the constraints as penalty terms. This can be done by rearranging and squaring the first constraint, and by noticing that the sum in the second constraint is non-negative, such that we can write a corresponding penalty term as a product, giving

$$x = \text{argmax}_x \left[\sum_{(i,j)\in E} x_{ij} \log c_{ij} - M_1 \sum_{i\in V} \left(\sum_{j,(i,j)\in E} x_{ij} - \sum_{j,(j,i)\in E} x_{ji} \right)^2 - M_2 \sum_{i\in V} \sum_{j,(i,j)\in E} x_{ij} \left(\sum_{j,(i,j)\in E} x_{ij} - 1 \right) \right],$$

where M_1 and M_2 are positive penalty coefficients. After solving the problem using a quantum annealer, we evaluate the resulting near-best x's first to ensure that they correspond to a feasible cycle (i.e., that the second and third terms'

values are zero), and then to check whether they correspond to a valid arbitrage opportunity (i.e., that the first term is positive).

Notes:

1. The number of binary variables in the optimization problem is equal to the number of edges $|E|$. If any asset can be converted into any other asset directly, the number of edges is $|E| = N(N-1)$, which is the number of ordered pairs for N assets. In this special case the problem is complete, but it can be much sparser in general. It is convenient to introduce the edge density d such that the number of variables/edges can be written as $|E| = dN(N-1)$ (when $d = 1$, the problem is complete).

2. The quantum annealer has a limited parameter range, and if the coefficients are outside of this range, they are scaled back into it. As the range of the problem parameters increases, the scaling factor increases, which forces an increasing number of the parameters into the noise range, which in turn leads to decreasing performance. Therefore, it is desirable that the coefficients of the first term depend on the conversion rate logarithmically

3. The optimum of this optimization problem always has a non-negative objective function value. This is evident by noticing that all terms are zero if all x's are set to zero, so any cycle with a negative objective function value can always be improved to zero by setting all of the respective x's to zero [3].

4. The above formulation does not forbid solutions that correspond to a collection of disjoint cycles. Hence, solutions must be checked and possibly split if they correspond to more than a single cycle.

5. In some cases, such as in spread trading, it is more convenient to maximize the dollar profit and work with a cost per edge instead of a conversion cost. In this case, the first term in the optimization problem can be written as the sum of costs $\sum x_{ij} c_{ij}$.

3 Formulation (node-based)

In certain cases, such as in futures trading, it might be of interest to find the best arbitrage opportunity of a particular structure/strategy. In addition, there are cases in which it is beneficial to consider a more general form of arbitrage, represented by a closed walk. A closed walk is a sequence of directed edges such that following the sequence from any node in the closed walk brings us back to the same node. It allows passing through the same node multiple times, whereas a cycle does not.

To this end, and inspired by Lucas [1], we define a binary variable $z_{i,k}$ for each node i, for each position k in a prospective closed walk of length K (where $K \leq N$ and N is the number of nodes/assets, as before). The optimization problem becomes

$$\begin{aligned} &\max \sum_{i,j \in V} \sum_{k=1}^{K} z_{i,k} z_{j,k+1} \log c_{ij} \\ &\text{s.t.} \sum_{i=1}^{N} z_{i,k} = 1 \quad \text{for all } 1 \leq k \leq K, \\ &\quad z_{i,k} z_{j,k+1} = 0 \quad \text{for all } (i,j) \notin E \text{ and } 1 \leq k \leq K, \end{aligned}$$

where we define $z_{i,K+1} = z_{i,1}$ (such that the first term includes the edge connecting the node in the last position to the node in the first position). The first constraint requires that each position in the closed walk have exactly one node assigned to it, and the second constraint requires that two adjacent nodes in the closed walk have an edge between them.

This optimization problem can be converted into a QUBO problem by writing appropriate penalty terms for each

constraint, giving

$$z = \text{argmax}_z \left[\sum_{i,j \in V} \sum_{k=1}^{K} z_{i,k} z_{j,k+1} \log c_{ij} - A \sum_{k=1}^{K} \left(\sum_{i=1}^{N} z_{i,k} - 1 \right)^2 - B \sum_{k=1}^{K} \sum_{(i,j) \notin E} z_{i,k} z_{j,k+1} \right].$$

Notes:

1. The number of binary variables in this optimization problem is equal to the number of nodes (assets) times the length of the closed walk, $N \times K$. Comparing with the number of variables required in the node-based formulation, $dN(N-1)$, we find that for a given problem, the second formulation will require fewer variables if $K < d(N-1)$. For the complete case $d = 1$, so the node-based formulation will require more variables only for $K = N$. For sparse graphs with $d \ll 1$, the node-based formulation requires more variables for most values of K.

2. Although this formulation might require more qubits than the previous formulation (depending on the K chosen; see above), and requires choosing the length of the desired closed walk in advance, it has some advantages. For example, it is possible to define a strategy that requires passing through particular nodes. An investor who has cash in USD might prefer an arbitrage opportunity that involves USD. This could be accomplished by fixing the variables that correspond to position i such that they are all zero except for the one that corresponds to USD, which would be fixed to one. Another possibility is to define a certain pattern and look for strategies that follow that pattern. For example, in spread trading it might be useful to define a pattern that requires passing through a strip multiple times, once for each of the constituents. This could be accomplished by adding a term like $-C \sum_{k=1}^{K} (z_{i,k} - z_{j_k})^2$, which would force the variables that correspond to position i to be equal to the variables that correspond to position j.

3. This formulation can be easily adapted to allow only cycles, by adding a constraint that each vertex can appear at most once in the cycle, $\sum_{k=1}^{K} z_{i,k} \leq 1$ for all $i \in V$, which can be accomplished by adding a penalty term of the type $-D \sum_{i \in V} (\sum_{k=1}^{K} z_{i,k})(\sum_{k=1}^{K} z_{i,k} - 1)$.

4 Extensions

It is possible to add a notion of risk to the optimization problem such that the optimal arbitrage opportunity strikes a balance between profit and risk. Risk in this context might take the form of price change prior to execution, for instance. For each conversion from asset i to asset j we can define a risk metric d_{ij}. The formulation would then have an added term which penalizes the total risk of the arbitrage opportunity,

$$-\gamma \sum_{(i,j) \in E} x_{ij} d_{ij},$$

where γ is the risk aversion, a coefficient used to weight the importance of the risk metrics. After calling the quantum annealer, we would first filter out all of the infeasible solutions and non-profitable (but feasible) arbitrage opportunities, and then order the profitable opportunities by their combined profit risk score, which is given by the sum of the objective function term and the new risk term.

Alternatively, it might be desirable to limit the length of the cycles found. Long cycles tend to carry more risk, as each edge commonly represents a separate transaction with its own risk of failure or price change. This could be done by penalizing long cycles using a linear or quadratic penalty term, such as

$$-M_3 \sum_{(i,j) \in E} x_{ij} \quad \text{or} \quad -M_4 \left(\sum_{(i,j) \in E} x_{ij} \right)^2.$$

The linear term could be viewed as including fixed transaction costs, which scale with the number of transactions.

5 An example

As a proof of concept, we used the 1QBit Software Development Kit (SDK) to implement the edge-based formulation from Section 2. Figure 1 shows an example with five currencies, all of which can be converted into each other, giving twenty conversion rates on a complete graph. We solved this example problem using both an exhaustive solver and the D-Wave 2X quantum annealer, both of which returned the same, optimal answer, as well as additional, profitable, near-best arbitrage opportunities. The most profitable arbitrage opportunity in this specific example involved four assets and provided a potential gain of 0.074%.

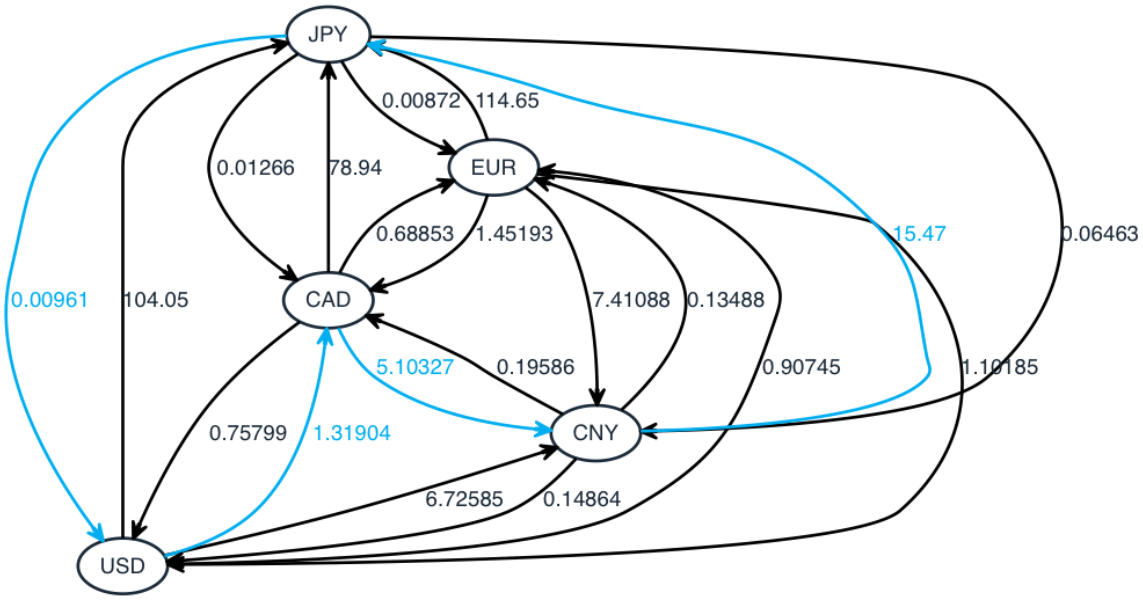


Figure 1: An example asset and conversion rate graph. The best (most profitable) arbitrage opportunity, marked in blue, involved four assets and showed a potential gain of 0.074%.

6 Usage in practice

The algorithm described above can be run on any graph of assets where the profit from traversing a cycle can be calculated by summing the weights of the edges. Unlike many graph theory algorithms, which terminate once a negative cycle is found, it is able (in principle) to find the best arbitrage opportunities. It can also force a set of edges to be traversed as part of the solution, which allows more-complex strategies to be represented. The algorithm has practical applications in futures and options trading, and in other markets where spreads and strategies play a significant role. Foreign exchange is one of many possible examples.

For the current generation of quantum annealers, a single call takes at least tens of milliseconds. This rules out its use on short-lived opportunities, which arise in highly liquid markets (such as crude oil or gasoline futures). However, these are often inaccessible even to very-high-speed traders, since the time to detect and respond to them is longer than the time that the market itself takes to correct them.

There are many applications that are well-matched to the speed of the current hardware. For example, the calculation of arbitrage-free settlement prices in a large set of thinly traded products would benefit from a quick and versatile arbitrage detector. There are also many less-liquid markets in which the detection of mispriced instruments could bring real value to the informed trader.

Acknowledgements

The author would like to thank Clemens Adolphs, Elham Alipour, Phil Goddard, Andrew Milne, Natalie Mullin, and Maxwell Rounds for their useful comments, and Marko Bucyk for editing the manuscript.

References

[1] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2(5), 2014.

[2] Robert Sedgewick. *Algorithms in C++*, volume 5. Pearson Education India, 2003.

[3] Wanmei Soon and Heng-Qing Ye. Currency arbitrage detection using a binary integer programming model. *International Journal of Mathematical Education in Science and Technology*, 42(3):369–376, 2011.