RESEARCH-ARTICLE

Lookalike Clustering for Customer Segmentation: a Comparative Study of Quantum Annealing and Classical Algorithms

BENEDETTA FERRARI, University of Modena and Reggio Emilia, Modena, MO, Italy

GIUSEPPE GNOCCHI

MANUEL IORI, University of Modena and Reggio Emilia, Modena, MO, Italy

SIMONE MASCARO, University of Modena and Reggio Emilia, Modena, MO, Italy

MIRKO MUCCIARINI, University of Modena and Reggio Emilia, Modena, MO, Italy

LUCA RINALDI

View all

Published: 11 August 2025

Citation in BibTeX format

GECCO '25 Companion: Genetic and Evolutionary Computation Conference Companion
July 14 - 18, 2025
Malaga, Spain

Conference Sponsors:
SIGEVO

https://doi.org/10.1145/3712255.3734355
ISBN: 9798400714641

Lookalike Clustering for Customer Segmentation: a Comparative Study of Quantum Annealing and Classical Algorithms

Benedetta Ferrari
University of Modena and Reggio Emilia
Reggio Emilia, Italy
benedetta.ferrari@unimore.it

Giuseppe Gnocchi
Credem S.p.A.
Reggio Emilia, Italy
ggnocchi@credem.it

Manuel Iori
University of Modena and Reggio Emilia
Reggio Emilia, Italy
manuel.iori@unimore.it

Simone Mascaro
University of Modena and Reggio Emilia
Reggio Emilia, Italy
309591@studenti.unimore.it

Mirko Mucciarini
University of Modena and Reggio Emilia
Reggio Emilia, Italy
mirko.mucciarini@unimore.it

Luca Rinaldi
Credem S.p.A.
Reggio Emilia, Italy
lrinaldi@credem.it

Gianluigi Salerno
Credem S.p.A.
Reggio Emilia, Italy
gsalerno@credem.it

Vittorio Tartarini
Credem S.p.A.
Reggio Emilia, Italy
vtartarini@credem.it

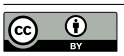
Andrea Vezzani
Credem S.p.A.
Reggio Emilia, Italy
a.vezzani@credem.it

Abstract

Customer segmentation is strategic for increasing business revenues, as it allows a company to offer tailored products and services. Companies have access to vast amounts of information to generate customer clusters. In the banking sector, for instance, transactional data may reveal several hidden customer characteristics. The lookalike clustering problem aims to expand an initial cluster by detecting similar customers from a larger set. In this work, we model the lookalike clustering problem as a Quadratic Unconstrained Binary Optimization problem and solve it by means of a Quantum Annealer (QA). We tune the penalization parameters of the QA through an adaptive subgradient-inspired algorithm. We then compare QA performance against those obtained by a Simulated Annealing and the classical integer programming solver Gurobi. The results show that QA, when properly tuned, is able to find the optimal solution for small instances of the problem, even when the number of variables exceeds the number of couplers of QA. It fails, instead, for larger ones, due to the high level of noise. The results also show that QA requires a very short time to generate feasible and optimal solutions compared to classical algorithms, thus proving its potential for the solution of fully connected graphs.

CCS Concepts

• **Hardware → Quantum technologies**; • **Mathematics of computing → Combinatorial optimization**; • **Applied computing → Marketing**.



GECCO '25 Companion, July 14–18, 2025, Malaga, Spain

ACM ISBN 979-8-4007-1464-1/2025/07
https://doi.org/10.1145/3712255.3734355

Keywords

Lookalike Clustering, QUBO, Quantum Annealer, Quantum Computing, Bank

ACM Reference Format:
Benedetta Ferrari, Giuseppe Gnocchi, Manuel Iori, Simone Mascaro, Mirko Mucciarini, Luca Rinaldi, Gianluigi Salerno, Vittorio Tartarini, and Andrea Vezzani. 2025. Lookalike Clustering for Customer Segmentation: a Comparative Study of Quantum Annealing and Classical Algorithms. In *Genetic and Evolutionary Computation Conference (GECCO '25 Companion), July 14–18, 2025, Malaga, Spain.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3712255.3734355

1 Introduction

In product and service industries, clustering customers with respect to common traits is often a crucial task to maximize marketing efficiency. Personalized offerings can positively influence customer buying behavior and brand loyalty. Several approaches have been attempted to cluster customers in different business sectors [2]. In the financial sector, this is particularly critical for customizing business proposals, improving portfolio management, and improving communication with existing or potential clients. Our research starts from the case study of Credito Emiliano S.p.A. (CREDEM), a credit institution based in Reggio Emilia (Italy) and having more than 400 locations in 19 Italian regions. CREDEM is established as a domestic commercial bank, but also provides side services, such as leasing, insurance, corporate finance, and wealth management, to a wide customer base. Within CREDEM, the Customer Analytics team is responsible for maximizing the efficiency of customer contacts. To this aim, the team has started developing a clustering algorithm to better promote its products and increase loyalty.

Given the difficulty in using and interpreting artificial intelligence approaches, hybrid clustering pipelines can be developed to create reliable clusters of similar customers. First, given a trait that we are interested in, an initial set of customers that are proved through data analysis to satisfy such a trait is included in an initial

seed cluster. However, data may not be informative or complete for some customers, leading to an imprecise or incomplete initial clustering. In these cases, given a common distance metric, it is useful to understand which customers are close enough to be inserted into the cluster (because of their small distance to the other customers of the cluster) but have been excluded as they do not satisfy the initial selection constraints. This is achieved through the solution of the so-called *lookalike problem*. Lookalike problems are employed, for example, by social media algorithms that propose advertisements based on the characteristics of user profiles [15]. They are usually solved through machine learning algorithms that evaluate the similarity of customers in a pool with the customers in the initial cluster [4].

In this work, we model the lookalike problem as an optimization problem based on a distance-based metric that evaluates the similarity between customers that have not yet been labeled and customers that have already been included in the initial seed cluster. Being the problem NP-hard, we chose to address it using quantum optimization. Specifically, we employed quantum annealing, an algorithm designed to run on a specialized class of quantum computers called Quantum Annealers (QA), which promises significantly better scalability than classical computers by exploiting quantum properties such as superposition and entanglement [10]. In QA, the objective function of the optimization problem corresponds to the total energy of the physical system, which can be represented by a Hamiltonian function $\mathcal{H}$. The goal is to find the lowest energy state of the system, $min(\mathcal{H})$. In order to be processed by a QA, an optimization problem has to be formulated as a Quadratic Unconstrained Binary Optimization (QUBO) problem. Specifically, each binary decision variable of the QUBO is mapped to a physical qubit, whereas coefficients in the objective function correspond to couplings between qubits. The problem is then solved by the physical process of QA, by gradually enforcing changes in the physical state so as to settle the system to a state of low energy, which corresponds to an optimal solution.

In our research, we use QA to solve a lookalike problem with the aim of expanding an initial seed cluster of customers so as to reach a final cluster having a target size and containing customers with similar characteristics. The results of QA are then compared with those obtained by a Simulated Annealing (SA) algorithm [8] and by a commercial Mixed Integer Linear Programming (MILP) solver. QAs are already available and accessible for research purposes [13] [14]. Their current state-of-the-art development has reached more than 5000 qubits and 15 interconnections (Pegasus topology) or more than 500 qubits and 20 interconnections (Zephyr topology) [7], but it is still unfortunately characterized by a large noise. QAs are particularly suited for problems with sparse matrices, where variables, and hence qubits, need few connections. Our problem is instead fully connected, as we evaluate the distance metric between each pair of customers within and outside the cluster. This prevents us from solving large instances of the lookalike problem.

The main contributions of this research are: (i) we study a relevant customer segmentation problem arising in the bank sector; (ii) we provide a QUBO formulation for the lookalike problem that we used to obtain a relevant segmentation; (iii) we propose a subgradient-inspired adaptive algorithm to set the penalization terms of the QA; (iv) we solve the problem with QA and compare its performance with that of classical optimization algorithms by means of extensive computational tests.

2 Problem Description

The lookalike problem aims to expand a defined initial seed cluster of points, corresponding to customers in our application, by including similar points. More in detail, we are given m points $i \in I$, each described by a set of k features. The distance d_{ij} between any pair of points $i, j \in I$ is computed as the Euclidean distance (as described in Section 4.1). In the customer segmentation domain, the features correspond to business variables defining customer behavior, such as the number of purchases or the purchase consistency. Given these features, a seed cluster can first be identified with a set of points $I_0 \subset I$ that satisfy deterministic conditions defined on their features. For instance, we can identify and group similar customers by looking at the expense types over subsequent months. Our aim is to expand the initial seed cluster by enforcing the selection of T points in the set I, including those in I_0, with the goal of minimizing the total distance d_{ij} between points within the final cluster.

Figure 1 represents our optimization problem on a small-scale example. The example contains six points, three of which classified as known-trait customers in I_0 defining the initial seed cluster, and three other candidate points represented with circles and corresponding to potential customers to be added to the cluster. The edges depicted in black represent the distances between points inside I_0, while the dashed and solid red lines represent the distances between points in I_0 and candidate points or between pairs of candidate points, respectively.

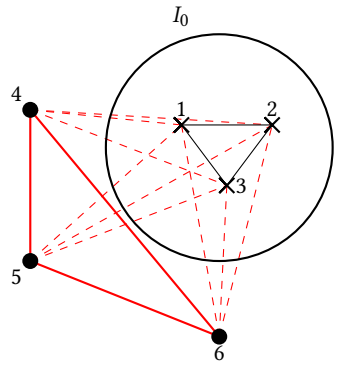


Figure 1: Representation of the lookalike problem on a small-scale example. The graph shows in I_0 the customers $\{1, 2, 3\}$, which are already part of the cluster, and outside customers $\{4, 5, 6\}$, which are candidates for the expansion of I_0

The problem is first formulated as a Binary Quadratic Problem (BQP) by using a binary variable x_i that assumes value 1 if point $i \in I$ is inside the final cluster, 0 otherwise. The variable is set to 1 for each point $i \in I_0$. The BQP model becomes:

$$\min(z) = \sum_{i \in I} \sum_{j \in I} d_{ij} x_i x_j \tag{1}$$

$$s.t. \quad x_i = 1 \qquad \forall i \in I_0 \tag{2}$$

$$\sum_{i \in I} x_i \geq T \tag{3}$$

$$x_i \in \{0,1\} \qquad \forall i \in I \tag{4}$$

The objective function (1) minimizes the sum of the distances between the selected points. Constraints (2) create the initial cluster, while constraint (3) imposes the selection of at least T points. Given the minimization aim of the problem and the fact that all d_{ij} values are assumed to be non-negative, the latter could be replaced by equality constraints, to simplify the translation to a QUBO, as described in the following section. Finally, constraints (4) express the binary nature of the variables.

2.1 QUBO formulation

One way to solve an optimization problem through the QA is to formulate it as a QUBO. A QUBO problem aims to find a binary vector x that minimizes a quadratic function $f(x) = x^T Q x$, with the Q-matrix representing the coefficients of the linear and quadratic terms of the function. We derived a QUBO formulation from the BQP by relaxing constraints (2) and (3) and adding them as potential penalizations in the objective function, through penalty coefficients λ_1 and λ_2, respectively. It should be noted that for binary decision variables, the relation $x_i = x_i^2$ always holds, and thus we can rewrite constraints (2) and (3) as follows.

Constraints (2) are relaxed into the following penalization:

$$\lambda_1 \sum_{i \in I_0} (x_i - 1)^2 = \lambda_1 \left(x_1^2 + \ldots + x_n^2 - 2x_1^2 - \ldots - 2x_n^2 + n \right)$$

$$= \lambda_1 \left(\sum_{i \in I_0} x_i^2 - 2 \sum_{i \in I_0} x_i^2 + n \right) = -\lambda_1 \sum_{i \in I_0} x_i^2 + \lambda_1 n \tag{5}$$

Constraints (3) become instead:

$$\lambda_2 \left(\sum_{i \in I} x_i - T \right)^2 = \lambda_2 (x_i^2 + \ldots + x_m^2 + 2x_1x_2 + \ldots + 2x_{m-1}x_m + T^2 +$$

$$- 2Tx_1^2 - \ldots - 2Tx_m^2) = \lambda_2 \left(\sum_{i \in I} x_i^2 + 2 \sum_{i \in I} \sum_{\substack{j \in I \\ j > i}} x_i x_j + T^2 - 2T \sum_{i \in I} x_i^2 \right) =$$

$$= \lambda_2 \left(\sum_{i \in I} \sum_{\substack{j \in I \\ j = i}} x_i x_j + \sum_{i \in I} \sum_{\substack{j \in I \\ j \neq i}} x_i x_j + T^2 - 2T \sum_{i \in I} x_i^2 \right) =$$

$$= \lambda_2 \sum_{i \in I} \sum_{j \in I} x_i x_j + \lambda_2 T^2 - 2\lambda_2 T \sum_{i \in I} x_i^2 \tag{6}$$

Formally, given (5) and (6) and noting that the constant terms $\lambda_1 n$ and $\lambda_2 T^2$ can be omitted in an optimization problem, the QUBO formulation of the lookalike problem is

$$\min(\mathcal{H}) = \sum_{i \in I} \sum_{j \in I} \left(d_{ij} + \lambda_2 \right) x_i x_j - \lambda_1 \sum_{i \in I_0} x_i^2 - 2\lambda_2 T \sum_{i \in I} x_i^2 \tag{7}$$

$$x_i \in \{0,1\} \qquad \forall i \in I \tag{8}$$

The Q matrix corresponding to the QUBO formulation (7)-(8) of our small-scale example is shown below, with $(a) = -(\lambda_1 + 2\lambda_2 T)$ and $(b) = -2\lambda_2 T$. In the upper left part, we highlight the coefficients of the three points in I_0, while in the lower right part we highlight those of all the external points. The white part of the matrix represents the links between internal and external elements.

$$Q = \lambda_2 + \begin{bmatrix} (a) & d_{12} & d_{13} & d_{14} & d_{15} & d_{16} \\ d_{12} & (a) & d_{23} & d_{24} & d_{25} & d_{26} \\ d_{13} & d_{23} & (a) & d_{34} & d_{35} & d_{36} \\ d_{14} & d_{24} & d_{34} & (b) & d_{45} & d_{46} \\ d_{15} & d_{25} & d_{35} & d_{45} & (b) & d_{56} \\ d_{16} & d_{26} & d_{36} & d_{46} & d_{56} & (b) \end{bmatrix} \tag{9}$$

2.2 Reduced Problem

We now propose a reduced formulation of the problem that limits the number of variables and consequently the number of qubits necessary to represent the problem. In the reduced problem, the points $i \in I_0$ are implicitly selected in the final cluster and are thus omitted from the problem. To this aim, we define a new set $I' = I \setminus I_0$ representing the residual set of points in the formulation, corresponding only to the candidate points. Hence, given the binary variable x_i defined only for each $i \in I'$, the objective function (1) can be divided into three terms representing the different connections in Figure 1:

$$\min(Z) = \sum_{i \in I_0} \sum_{j \in I_0} d_{ij} + \sum_{i \in I'} \sum_{j \in I_0} d_{ij} x_i + \sum_{i \in I'} \sum_{j \in I'} d_{ij} x_i x_j \tag{10}$$

The leftmost term of (10) corresponds to the sum of the distances between the points inside I_0. This is a constant, and hence can be removed from the formulation. Similarly, in the second term, for each point $i \in I'$ the sum of the distance with all the points in I_0 is a constant and can be pre-computed as $B_i = \sum_{j \in I_0} d_{ij} \forall i \in I'$. Finally, note that constraints (2) can be removed, so the BQP (1)-(4) can be rewritten as follows:

$$\min(Z) = \sum_{i \in I'} B_i x_i + \sum_{i \in I'} \sum_{j \in I'} d_{ij} x_i x_j \tag{11}$$

$$s.t. \sum_{i \in I'} x_i \geq T' \tag{12}$$

$$x_i \in \{0,1\} \quad \forall i \in I' \tag{13}$$

where $T' = T - |I_0|$ is the additional number of points to be added to the cluster.

The equivalent QUBO formulation is then:

$$\min(\mathcal{H}_{\mathcal{R}}) = \sum_{i \in I'} \sum_{j \in I'} (d_{ij} + \lambda) x_i x_j + \sum_{i \in I'} (B_i - 2\lambda T') x_i^2 \tag{14}$$

$$x_i \in \{0, 1\} \quad \forall i \in I' \tag{15}$$

where a unique penalization term λ is present.

3 Adaptive algorithm for determining λ

A critical aspect of the QUBO formulation is the tuning of the penalization coefficient λ. It should be a positive constant to provide the right penalization in a minimization problem. However, its magnitude is difficult to estimate: on one side a too low value may produce infeasible solutions, as the penalization for violated constraints is not strong enough; on the other hand a too high value may result in suboptimal solutions because of the flattened search space. An effective but computationally expensive approach to estimate λ is through trial and error, attempting to infer the behavior of the objective function. Similarly, incremental values of λ can be tested iteratively on the problem until a stopping criterion is met. For instance, an algorithm could stop when the number of infeasible solutions generated with a specific λ is lower than 20%, as in [5]. However, this approach can be computationally expensive or even suboptimal if the increasing step is too high.

In this work, we implemented a subgradient-inspired adaptive algorithm to set the value of λ. The algorithm allows for an automatic adaptation of λ that gradually converges to the best value. A pseudocode is shown in Algorithm 1.

At the beginning, λ is set to 1. Then its value is updated by a step of value α during the execution of the algorithm. More in detail, at each iteration λ is used to generate the current solution x, which is then evaluated to state its feasibility and determine the value of the objective function z, after deduction of possible penalties. If the solution is not feasible, λ is increased. Otherwise, the algorithm evaluates two consecutive scenarios:

(1) if a new feasible solution has been found after an infeasible one, the step α is reduced to allow a fine tuning of λ;
(2) if a new best solution z^* is found or if the new solution is higher than z^*, λ is decreased.

During the execution, list_λ^* keeps track of the different values of λ that give the same optimal solution.

When the number of iterations without improvements ρ becomes equal to a predefined threshold τ, then α is decreased. The algorithm stops when the maximum number N of iterations is reached or $\rho \geq \tau$ and $\alpha \leq \alpha_{min}$. Finally, the best value of λ is computed as the average value of list_λ^*.

All the algorithm parameters (α_0, α_{min}, γ, and τ) have been manually tuned in a preliminary set of computational experiments. However, the solution speed and convergence could benefit from further and more methodic parameter optimization.

Algorithm 1 Subgradient-inspired Adaptive Algorithm

```
α0 ← 3.5, α ← α0, α_min ← 0.05
λ ← 1, γ ← 0.4, τ ← 6, ρ ← 0
while iter < N do
    x ← Solve (λ)                                   ▷ Solve the QUBO problem
    is_feasible, z ← Evaluate (x)
    if not is_feasible then
        λ ← λ + α                                   ▷ Increase λ to speed up convergence
        ρ ← 0
        first_feasible ← false
    else
        if not first_feasible then                  ▷ Feasible solution found after infeasible one
            first_feasible ← true
            α ← γ · α                               ▷ Reduce step size for fine-tuning
        end if
        if z < z* then                              ▷ New best feasible solution
            z*, λ* ← z, λ
            list_λ* ← [λ*]
            ρ ← 0
            λ ← λ − α                               ▷ Try to improve by reducing λ
        else
            ρ ← ρ + 1
            if z > z* then
                λ ← λ − α                           ▷ Step back if we got worse
            else
                Add λ to list_λ*
            end if
            if ρ ≥ τ then                           ▷ Check number of no improvements
                α ← max(α_min, γ · α)               ▷ Reduce step size
                ρ ← 0
                if α ≤ α_min then
                    break                           ▷ Converged to minimum step size
                end if
            end if
        end if
    end if
end while
return Mean (list_λ*), z*
```

4 Computational Results

We performed several tests to evaluate the performance of the QA on various instances of the lookalike problem. Specifically, the experiments were carried out through the D-Wave QA and the results were compared with classical approaches. First, we exploited the implementation of the classical SA available in D-Wave through the library *neal*. The SA, while following the same physical principle of a quantum annealer, does not suffer from noise that reduces the precision of the solution and does not have limitation in the dimension of the instances to be solved as there is no real physical implementation. Moreover, as it is a heuristic algorithm, it can be easily implemented as a solution method by companies without the need of paying a commercial fee. The BQP formulation of the reduced lookalike problem has also been solved by the commercial solver Gurobi v12.0.0, letting it run under default conditions. All algorithms were implemented in Python and executed on a Windows 10 Pro N Intel(R) Xeon(R) Gold 6430 with 16GB of equipped RAM.

4.1 Test Set

The proposed algorithms were first tested on financial datasets provided by CREDEM. However, for privacy reasons, it is not possible to make the results public. Therefore, we decided to generate several instances starting from an online dataset dedicated to supervised clustering of customers [12]. The dataset is composed of demographic and purchasing behavior data from more than 300.000 customers, including age, income, marital status, education, and spending habits, such as product preferences, campaign responses, and purchase frequency. From this dataset, we selected only the features that are relevant to customer classification and indicative of individual spending behavior. Specifically, we retained the customer income, the recency (number of days since the last purchase),

the amount of money spent across six product categories (e.g., wine, meat, fish), and the number of purchases made through different sales channels (e.g., web, store, catalog). After feature selection, we standardized the data using the standard scaler method, which transforms each feature by subtracting the mean and scaling to unit variance. This preprocessing step ensures that all variables contribute equally to the analysis, avoiding biases due to different scales or units of measurement. We then computed the pairwise Euclidean distance matrix and applied multidimensional scaling to represent the high-dimensional data in a lower-dimensional space while preserving their pairwise distances.

After that, we generated two clusters through the hierarchical clustering algorithm available in the *sklearn* library. The points in I were selected among the points near the centroids of the two clusters. Then, the points in I_0 were selected among the points in a single cluster and the number T' of accepted candidates was set to be equal to the number of points in I' that belong to the same cluster as the point I_0. In this way, we can easily evaluate the goodness of the solution generated by the three optimization approaches even in the cases in which the problem is not solved to proven optimality. Finally, the Euclidean distance matrix was used to compute the B_i coefficients for each $i \in I'$.

The resulting test set is composed of 72 instances with a varying number of points $|I'| = \{4, 8, 16, 32, 64, 80\}$. The first three values correspond to graphs that fit the QA topology without need of qubit chains. The last three, instead, exceed quantum hardware, with 80 qubits approaching the QA size limit given the chains needed to connect each pair of qubits. The expansion demand T' is then set as a percentage of $|I'|$, that is, $T' = \{25\%, 50\%, 75\%\} \cdot |I'|$. We also varied the number of points in the initial seed cluster such that $|I_0| = \{\frac{1}{4}, 1, \frac{3}{2}, 4\} \cdot |I'|$. The entire set of instances is publicly available at https://github.com/regor-unimore/Lookalike-Clustering-for-Customer-Segmentation.

4.2 Setting of λ

The proposed subgradient-inspired adaptive algorithm (Algorithm 1) was implemented and tested with the SA of D-Wave, because SA allows a user to perform several tests with a fast execution time and without the need to use a quantum computer. We compared this approach against the fixed-step method described in [5].

Figures 2 and 3 show the evolution of the adaptive and fixed-step algorithms, respectively, for a single instance with $|I'| = 16$ and $T' = 8$. The instance size was chosen as it corresponds to the larger instance that can be embedded within a Zephyr architecture without using qubit chains. Both the approaches gradually converged to the optimal objective function (green dashed line). Moreover, the optimal λ obtained by both approaches are very similar (211.34 and 211.00, respectively). However, the fixed-step algorithm required 282.56 seconds to reach the final λ value and complete its execution, whereas the adaptive one converged in just 2.11 seconds.

A comprehensive comparison of the two algorithms is provided in Table 1. Each line of the table reports average values for 12 instances having same cardinality $|I'|$, namely: average λ value obtained at the end of the execution, average number of iterations (Iter), and average execution time in seconds (T[s]). Although both methods yield nearly identical λ values, the adaptive algorithm

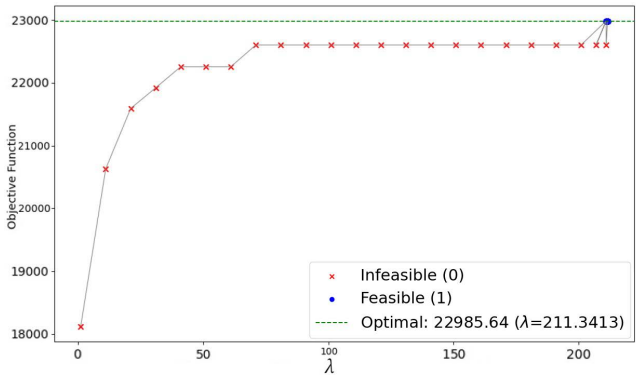


Figure 2: λ optimization with the adaptive algorithm. The first feasible solution is also the global optimum. Then, convergence of λ is achieved after a few iterations returning infeasible solutions

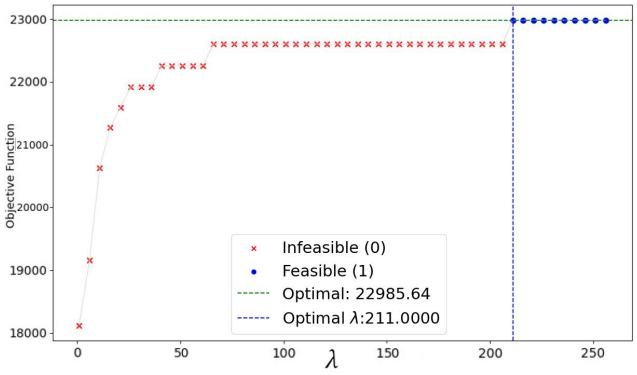


Figure 3: λ optimization with the fixed-step algorithm. The optimal λ is to the lowest value among subsequent ones resulting in the optimal solution

achieves an average runtime reduction of 98% with respect to the fixed-step one, despite a higher number of iterations. This is mainly due to the fact that the fixed-step algorithm requires a high number of calls to SA at each iteration (namely, 10 calls per iteration) to provide a good assessment of the feasibility of the QA solution which is expected to be obtained with that particular λ value. The adaptive algorithm requires, instead, only a single SA call per iteration.

Deepening the analysis on λ, we also noticed that its best value strongly depends both on T' and $|I_0|$, increasing as these values grow. This trend is visible in Figure 4, which shows variations of λ across instances with $|I'| = 16$.

Notably, the optimal λ value found via SA also performs well for QA, despite the latter being affected by noise. Our tests show that, with the default QA setup, achieving this performance requires a high number of reads. Figure 5 presents the results of the adaptive algorithm using QA with 3000 reads. The first feasible point was found with a value of λ lower than the optimal one, probably due to noise and oscillations inside the QA. The resulting λ value (204.83) is indeed very close to the value found via SA (211.34), further

Table 1: Comparison of algorithms performance for the determination of λ (average values over 12 instances per line)

	Fixed-Step [5]			Adaptive		
$\|I'\|$	λ	Iter	T[s]	λ	Iter	T[s]
4	13.08	12.42	5.32	13.78	21.25	0.09
8	26.83	15.17	13.90	26.99	23.42	0.21
16	52.67	20.33	40.57	52.94	25.50	0.49
32	111.83	32.17	158.86	112.33	30.75	1.50
64	224.33	54.67	817.43	226.15	43.17	10.46
80	284.75	66.75	1497.12	286.38	49.67	12.66

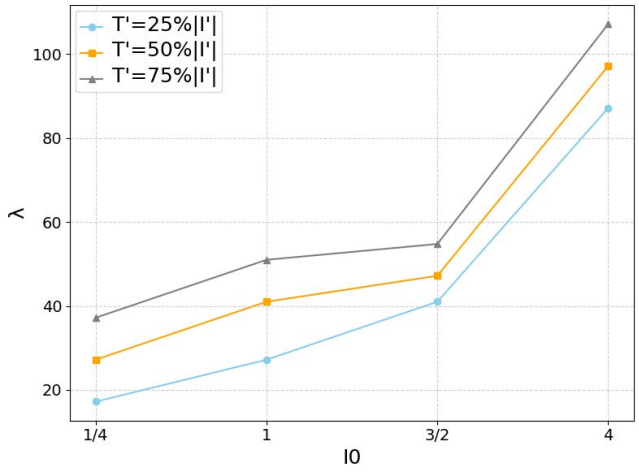


Figure 4: Best λ values for instances with $|I'| = 16$ obtained by varying T' and I_0. λ is positively related to both parameters, as they impact on the magnitude of coefficients in the Q-matrix

validating the effectiveness of the adaptive approach. However, as described in Section 4.3, fine-tuning the QA parameters can reduce the number of reads, thereby decreasing computation time while maintaining solution quality.

4.3 QA setup

The experiments were carried out on the D-Wave Zephyr QA, which offers a higher number of connections between neighboring physical qubits. Specifically, the Zephyr cell has 20 couplers and a nominal qubit length of 16, meaning that physically at most a 16 x 16 fully connected Q-matrix can be represented without the need for multiple physical qubits for each variable. With larger matrices, D-Wave has automatic algorithms that map the problem onto the hardware, creating qubit chains that can introduce further noise in QA. To improve the performance of the annealer for our problem, three parameters of the algorithm were discussed and carefully tuned: type of embedder, chain strength, and anneal schedule.

Embedder. The type of embedding on a QA significantly affects how the problem adapts to the physical system. We noticed that the default *Composite* embedder works well for fully-connected problems with fewer than 16 variables, but it fails to adjust to larger

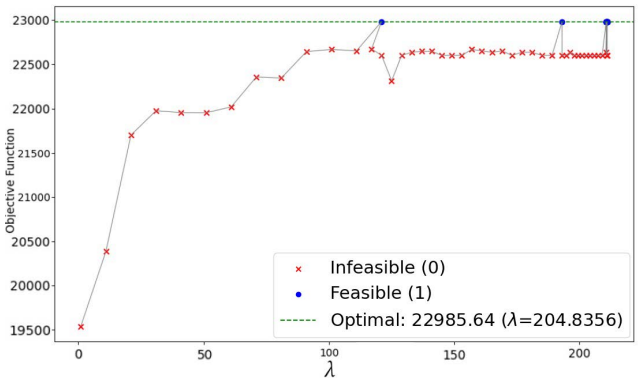


Figure 5: QA λ optimization with 3000 runs. The λ parameter found with the QA with a high number of reads is similar to the optimal λ found by SA, despite QA instabilities and noise

ones. In contrast, the *Clique* embedder proved to be more effective, as it is specifically suited to reduce the chain length and improve performance for complete graphs [3]. Therefore, we employed the *Clique* embedder for our tests.

Chain strength. Chain strength is a critical parameter for solving problems that exceed the topology of QA and therefore lead to the creation of qubit chains. In fully connected problems, chain breaks frequently occur, preventing finding optimal or even feasible solutions. When a chain breaks, a variable may assume multiple values (i.e., the qubits of the same chain, which should ideally have all the same value, have different values). To solve the issue, D-Wave systems implement an automatic fixer that returns, for each variable, the qubits value that appears most frequently among the qubits assigned to that variable. However, this approach often yields suboptimal solutions, and hence chain breaks should always be avoided. Proper tuning of the chain strength parameter is a key issue to achieving this.

The Ocean SDK documentation from D-Wave systems suggests to use the default chain-strength value calculated from the *uniform_torque_compensation()* function, which adjusts the value based on instance-specific characteristics and coefficients [6]. In our test set, the default values assigned by D-Wave increased with the number of qubits (i.e., with $|I'|$) but also with T' and $|I_0|$. Figure 6 shows an example of the chain strength values assigned by the Clique embedder for a single instance. The chain strength depends on the coefficients within the instance matrix and, therefore, varies with the value of λ.

It can be noticed from Figure 6 that the default chain strength values are insufficient to optimally solve the problem, as they generally produce infeasible solutions, or feasible ones with chain breaks. This behavior was common to practically all instances that we tested. Although previous attempts have been made to predict optimal chain strength, existing methods were not directly applicable to our study [11]. Therefore, we decided to manually tune the chain strength for each case, following the general guidelines of the D-Wave documentation, which suggests that too high values can

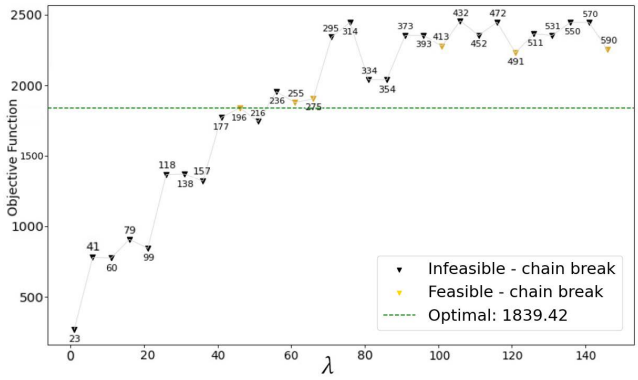


Figure 6: Example of default chain strength assigned by the Clique embedder for each λ tested on a given instance. The value of chain-strength is highly variable and related to the choice of λ, making the overall parameter setup complex

disrupt the QUBO formulation, while low values can cause severe chain break errors.

Anneal schedule. The anneal schedule was chosen following the work in [9]. There, the authors show that including a pause in a specific time-spot during a 16-qubit QA instance can dramatically improve the precision of the computer. Given the topology of the Zephyr computer, we adopted the anneal schedule [(0.0,0.0), (3.7,0.37), (13.7,0.37), (20.0,1.0)] as suggested in [9], which consists of 10μs anneal time and 10μs pause starting at 3.7μs. Using this method, we obtained relevant improvements in our computational results.

4.4 Lookalike algorithms comparison

The results of the different lookalike algorithms on the test set are presented in Table 2. To ensure consistency, we performed ten runs on both SA and QA on each instance, setting the number of reads (i.e., iterations) per run to 50. For each line we report the average results for instances with the same values of $|I'|$ and T' (i,.e., each line reports average values for four instances). The complete set of results is available at https://github.com/regor-unimore/Lookalike-Clustering-for-Customer-Segmentation.

For QA, we provide the percentage of feasible and optimal solutions obtained, the optimality gap, and the execution time. The optimality gap is computed only for the feasible solutions as the percentage difference between the QA and the Gurobi solution values, referred as Obj in the table. Since SA and Gurobi always found optimal solutions, we focus only on their computation time. The reported time corresponds to the operative execution of each algorithm: for QA, this includes the access time of the Quantum Processing Unit (QPU) (i.e., the sum of sampling and programming time), while for SA, it includes the total computing time on our system (i.e., the sum of pre-processing, sampling, and post-processing times).

QA consistently found the optimal solution for small problem instances with $|I'| \in \{4, 8\}$. In these cases, the number of variables remains below the number of couplers per qubit in the physical

Table 2: Algorithms performance

		QA				SA	Gurobi	
$\|I'\|$	T'	%Feas	%Opt	Gap	T[s]	T[s]	T[s]	Obj
4	25%	100%	100%	0%	0.02	0.00	0.03	97.40
4	50%	100%	100%	0%	0.02	0.00	0.03	113.39
4	75%	100%	100%	0%	0.02	0.00	0.03	126.19
8	25%	100%	100%	0%	0.02	0.01	0.03	384.54
8	50%	100%	100%	0%	0.02	0.01	0.03	462.28
8	75%	100%	100%	0%	0.02	0.01	0.03	565.26
16	25%	100%	100%	0%	0.02	0.02	0.03	1 610.98
16	50%	93%	93%	0%	0.02	0.02	0.03	1 886.30
16	75%	98%	98%	0%	0.02	0.02	0.04	2 237.78
32	25%	98%	98%	0%	0.03	0.04	0.04	6 554.70
32	50%	83%	83%	0%	0.03	0.05	0.04	7 643.70
32	75%	100%	100%	0%	0.03	0.04	0.04	9 009.98
64	25%	53%	0%	58%	0.03	0.12	0.06	25 962.44
64	50%	30%	0%	23%	0.03	0.14	0.04	30 298.66
64	75%	18%	0%	16%	0.03	*0.18	0.04	35 825.21
80	25%	10%	0%	84%	0.03	0.18	0.07	40 273.20
80	50%	40%	0%	13%	0.03	0.21	0.06	47 614.47
80	75%	0%	0%	–	0.03	0.18	0.06	55 873.50

*Optimal solutions for this subset were found by letting SA run for 100 iterations (instead of 50) for instances with $|I_0| \in \{1, \frac{3}{2}\}$

system, eliminating the need for qubit chains. Moreover, thanks to the fine-tuning of the parameter setting, we successfully solved some larger instances to proven optimality. This is a relevant result given the challenges posed by high noise in fully connected graphs, and also considering that we used a low number of reads. Specifically, for instances with $|I'| \in \{16, 32\}$, QA achieved a high ratio of feasible and optimal solutions, reaching 100% in two classes of instances. However, performance declines as $|I'|$ increases. When nearing full usage of the Zephyr QPU (563 qubits), with both 64 and 80 point instances, we observed a degradation of the optimal and feasible ratios, until no feasible solution was found for the larger instances. This could be related to errors in the *Clique* embedding algorithm due to faulty qubits or due to suboptimal values of chain strength [3]. A possible way to overcome this negative outcome would be to implement a quick repair heuristic that takes in input the infeasible solutions and obtains feasibility by either removing or adding points in the cluster. This repair heuristic would run on a common computer, and the resulting algorithm would benefit from both quantum and standard computers.

In terms of computational time, QA demonstrates high efficiency, achieving an average speedup of 40% compared to Gurobi. Notably, its performance remains stable as instance size increases, showing no significant slowdown. In contrast, while SA outperforms QA on very small instances, it becomes increasingly slower for larger ones, requiring six or seven times more computation time than QA. For these instances, however, SA is still able to find optimal solutions even when QA fails.

5 Conclusions

In this study, we proposed a QUBO formulation for the lookalike problem, which aims at finding the expansion of an initial cluster having given size and minimum sum of distances between the selected points. This optimization problem originates from a financial application at CREDEM bank, where it aids in customer segmentation for marketing and business improvement. The QUBO problem was solved using the QA from D-Wave. We carefully optimize its key parameters and proposed a subgradient-inspired adaptive algorithm for the tuning of its main penalization parameter λ. The computational results we obtained were compared with those obtained via SA and the commercial solver Gurobi.

Our computational results show that QA can effectively solve the lookalike problem to optimality, even when the number of variables exceeds the number of couplers of QA. This is particularly noteworthy given the fully connected nature of the problem, which typically presents challenges for QA. These findings highlight the potential of QA to tackle lookalike clustering problems.

Future research will focus on developing a repair algorithm to restore the feasibility of infeasible solutions that violate the constraints of the problem, similar to the approach proposed in [1]. Additionally, we aim to further investigate the correlation between λ, chain strength, and anneal schedule to refine performance. Finally, we are particularly interested in designing an improved embedding algorithm to optimize qubit allocation, reducing the use of resources and noise on the QPU. This would enable the reliable solution of larger problem instances, further advancing the practical applicability of QA for complex lookalike applications.

Acknowledgments

We would like to thank Officine Credem and Credem Bank not only for their financial support, but also for the trust, spaces, and services offered during this research.

References

[1] Thiago Alves de Queiroz, Manuel Iori, Alberto Locatelli, and Matthieu Parizy. 2025 (forthcoming). Solving the Cubic Knapsack Problem using the Quantum-Inspired Digital Annealer Technology. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

[2] Miguel Alves Gomes and Tobias Meisen. 2023. A review on customer segmentation methods for personalized customer targeting in e-commerce use cases. *Information Systems and e-Business Management* 21, 3 (2023), 527–570.

[3] Kelly Boothby, Andrew D. King, and Aidan Roy. 2020. Fast clique minor generation in Chimera qubit connectivity graphs. arXiv:1507.04774

[4] Anna Mariam Chacko, Bhuvanapalli Aditya Pranav, Bommanapalli Vijaya Madhvesh, and A. S. Poornima. 2021. Customer Lookalike Modeling: A Study of Machine Learning Techniques for Customer Lookalike Modeling. In *Intelligent Data Communication Technologies and Internet of Things*, Jude Hemanth, Robert Bestak, and Joy Iong-Zong Chen (Eds.). Springer Singapore, 211–222.

[5] Darren M Chitty, James Charles, Alberto Moraglio, and Ed Keedwell. 2024. Applying a Quantum Annealer to the Traffic Assignment Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 814–822.

[6] D-Wave Systems. 2025. *Chain Strength Documentation.* https://docs.dwavequantum.com/en/latest/ocean/api_ref_system/embedding.html#module-dwave.embedding.chain_strength Accessed: 2025-05-02.

[7] D-Wave Systems. 2025. D-Wave Quantum Systems. https://www.dwavesys.com/solutions-and-products/systems/ Accessed: 2025-04-02.

[8] Daniel Delahaye, Supatcha Chaimatanan, and Marcel Mongeau. 2018. Simulated annealing: From basics to applications. In *Handbook of metaheuristics*. Springer, 1–35.

[9] Neil G Dickson et al. 2013. Thermally assisted quantum annealing of a 16-qubit problem. *Nature communications* 4, 1 (2013), 1903.

[10] Hristo N. Djidjev, Guillaume Chapuis, Georg Hahn, and Guillaume Rizk. 2018. Efficient Combinatorial Optimization Using Quantum Annealing. *arXiv preprint arXiv:1801.08653* (2018).

[11] Valentin Gilbert and Stéphane Louise. 2024. *Quantum Annealers Chain Strengths: A Simple Heuristic to Set Them All.* Springer Nature Switzerland, 292–306.

[12] Miguel Jimenez. 2023. New Marketing Campaign. https://www.kaggle.com/datasets/mikejimenez24/new-marketing-campaign Accessed: 2025-05-02.

[13] Vaibhaw Kumar, Gideon Bass, Casey Tomlin, and Joseph Dulny. 2018. Quantum annealing for combinatorial clustering. *Quantum Information Processing* 17 (2018), 1–14.

[14] Kenichi Kurihara, Shu Tanaka, and Seiji Miyashita. 2014. Quantum annealing for clustering. *arXiv preprint arXiv:1408.2035* (2014).

[15] Tereza Semerádová and Petr Weinlich. 2019. Computer Estimation of Customer Similarity With Facebook Lookalikes: Advantages and Disadvantages of Hyper-Targeting. *IEEE Access* 7 (2019), 153365–153377.