A Quantum-Leap into Schema Matching: Beyond 1-to-1 Matchings

LUISA GERLACH, Humboldt-Universität zu Berlin, Germany
TOBIAS KÖPPL, Fraunhofer-Institut für Offene Kommunikationssysteme (FOKUS), Germany
STEFANIE SCHERZINGER, Universität Passau, Germany
NICOLE SCHWEIKARDT, Humboldt-Universität zu Berlin, Germany
RENÉ ZANDER, Fraunhofer-Institut für Offene Kommunikationssysteme (FOKUS), Germany

Schema matching refers to the task of identifying corresponding attributes of different database relation schemas to enable the efficient integration of the associated datasets. We model the task of finding suitable 1:N / N:1 global matchings in relational schemas as an optimization problem. We show that this optimization problem is NP-hard. We then translate the optimization problem into the problem of minimizing a particular rational-valued function on binary variables. The latter enables us to utilize modern quantum algorithms for solving the global matching problem, a crucial stage in schema matching. We also report on preliminary experimental results that serve as a proof-of-concept for our approach.

CCS Concepts: • **Theory of computation → Database theory**; **Quantum computation theory**; • **Information systems → Information integration**.

Additional Key Words and Phrases: schema matching, global matching, NP-hard optimization problem, minimization of a function on binary variables, QAOA: Quantum Approximate Optimization Algorithm

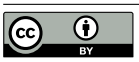
ACM Reference Format:
Luisa Gerlach, Tobias Köppl, Stefanie Scherzinger, Nicole Schweikardt, and René Zander. 2025. A Quantum-Leap into Schema Matching: Beyond 1-to-1 Matchings. *Proc. ACM Manag. Data* 3, 2 (PODS), Article 89 (May 2025), 24 pages. https://doi.org/10.1145/3725226

1 Introduction

Schema matching is a fundamental challenge in data integration that has been extensively researched for decades [5, 6, 10]. The schema matching problem arises whenever structured data from different sources need to be combined. Data scientists must then identify semantic correspondences between elements in the schemas. With relational databases, this involves matching attributes from one relation to those in another relation. Figure 1 shows an example, where semantically related data must be integrated from two relations, but the relations have different schemas.

Notably, the task of *schema matching* is related to, but different from, the task of *schema mapping*. Schema *matching* identifies semantic correspondences between schemas. In the aforementioned example, it appears natural to match the attributes "FirstName" and "LastName" from relation schema R_1 to the attribute "Name" of relation schema R_2. Once a schema matching is established, *schema mapping* reveals semantic relationships between schemas, such as the operations that

Authors' Contact Information: Luisa Gerlach, luisa.gerlach@hu-berlin.de, Humboldt-Universität zu Berlin, Berlin, Germany; Tobias Köppl, tobias.koeppl@fokus.fraunhofer.de, Fraunhofer-Institut für Offene Kommunikationssysteme (FOKUS), Berlin, Germany; Stefanie Scherzinger, stefanie.scherzinger@uni-passau.de, Universität Passau, Passau, Germany; Nicole Schweikardt, schweikn@hu-berlin.de, Humboldt-Universität zu Berlin, Berlin, Germany; René Zander, rene.zander@fokus.fraunhofer.de, Fraunhofer-Institut für Offene Kommunikationssysteme (FOKUS), Berlin, Germany.



ACM 2836-6573/2025/5-ART89
https://doi.org/10.1145/3725226

FirstName	LastName	Address	Birthday	ID
Anne	Rose	Domkloster 4, 50667 Köln	15.03.1977	001
Leon	Meyer	Oktavie-Allee, 66687 Wadern	24.12.1958	002
Charlotte	Frey	Maulbeerallee, 14469 Potsdam	09.08.1994	003

(a) Relation schema R_1 and an R_1-relation I_1

Name	StreetNo	City	DateOfBirth
Alice Alison	1402 Broadway Avenue J	Galveston	11/03/1972
Joline Shaw	8601 North Thorne Lane SW	Lakewood	04/23/1981
Bob Smith	80 Hesperus Ave	Gloucester	10/31/1966

(b) Relation schema R_2 and an R_2-relation I_2

Fig. 1. Running example

transfer data from one schema to the other. In the example from before, concatenating the values for the attributes "FirstName" and "LastName" in R_1 naturally maps to the attribute "Name" in R_2.

In practice, data scientists use professional software tools, such as *IBM Clio* [16, 24] and *Altova MapForce* [3], to interactively (but largely manually) match and map different schemas. To reduce labor-intensive manual efforts, schema matching algorithms aim to automate the task. They first identify potential semantic correspondences, represented by similarity scores, between all pairs of attributes. Over the years, many matching methods have been proposed [20, 27]. These methods differ in which information they rely on to compute these scores, such as attribute names, attribute values, or also external information, such as ontologies.

Once all pairs of attributes have been scored, a *global matching* is computed, i.e., a subset of the potential matches is selected to maximize some cumulative score. This paper addresses the problem of computing global matchings. Most existing approaches limit themselves to 1:1 cardinalities in matchings between relational attributes [18], often to ensure that solutions to the problem can be computed in polynomial time. When striving for 1:1 cardinalities, instances of the problem can traditionally be reduced to solving the stable marriage problem, or the maximum weight graph matching problem [10]. But striving only for 1:1 cardinalities also restricts the applicability of schema matching algorithms in real-world scenarios. E.g., to match the attributes for first and last names in one schema with a composite full-name attribute in another schema — which is a natural solution in the example shown in Figure 1 — higher-cardinality matches must be supported. In the literature, higher cardinalities are mainly studied in the form of 1:N or N:1 or N:M cardinalities (cf., [14, 27]). In particular, [14] discusses real-world scenarios and points out that it would be highly desirable to not a-priori restrict attention to fixed cardinalities, but to allow for flexible cardinalities based on the particular data present in the given relation schemas and relations.

Scope. In our work, we consider global matching beyond the typical 1:1 correspondences between attributes, and we allow for both 1:N and N:1 cardinalities within one matching (henceforth, we call this 1:N / N:1). We provide a new formalization based on a weighted bipartite graph, each node of which represents a single attribute of the considered relation schemas (in contrast to this, the models of [14] that go beyond 1:1 cardinalities require each node to represent a *set* of attributes, and thus involve a considerably larger number of nodes). As we show, for our problem definition, global matching becomes NP-complete to solve. To address this added challenge, we frame the problem as that of minimizing a rationally valued function on binary variables. This novel formulation allows us to resort to a *Quantum Approximate Optimization Algorithm* (QAOA) for the computation of solutions and, ultimately, to offload this computation to quantum hardware.

Quantum solutions for data management problems. In recent years, leading cloud providers such as *AWS*, *Google*, and *IBM*, have integrated quantum hardware into their infrastructure-as-a-service

portfolios. The availability of quantum computing technology to researchers and developers and the potentially exponential speed advantage over classical algorithms have sparked interest in exploring use cases for hybrid classical-quantum software solutions, which combine computations on classical hardware (CPUs and GPUs) and on quantum hardware (QPUs). In general, tasks that are computationally challenging yet which can be formulated compactly are promising candidates.

Within database systems research, several optimization problems have been investigated as candidate use cases. This includes query planning in multi-query optimization [11, 32], join reordering [30, 35], cardinality estimation [19], index selection [33], and transaction scheduling [8].

However, exploration from the field of data integration has been much more limited, with one contribution on entity matching [7], and one early proof-of-concept showcasing schema matching [13]. This early tool demo is limited to 1:1 matchings and reduces the problem of global matching to an instance of the stable-marriage problem, for which a quantum-based solution has already been established [29].

Our work pushes the boundaries of this emerging field by introducing a novel formulation for global matching which allows for offloading parts of the computation to quantum hardware and which supports 1:N / N:1 cardinalities in schema matchings. This is a technological leap forward in schema matching and might further open the field for quantum-enhanced data integration.

Contributions. The contributions of this paper are as follows:

(1) We propose a novel formalization of the global matching problem in schema matching. Our formulation allows for 1:N / N:1 cardinalities in matchings. We model the problem of finding the most suitable such matchings as an optimization problem which we call GMOP.
(2) We show that this optimization problem is NP-complete.
(3) We translate the optimization problem GMOP into the problem of minimizing a particular rational-valued function on binary variables. This allows for the problem to be solved using a QAOA algorithm and, consequently, also on a combination of classical and quantum hardware.
(4) In a micro-experiment, we apply an implementation of our approach to a collection of scenarios from a schema matching benchmark. We show that, within the limitations of today's quantum simulators, we can indeed solve schema matching using a hybrid quantum-classical algorithm. This serves as an early proof of concept that motivates further research in this field.

Organization. The remainder of this paper is organized as follows. Section 2 provides the necessary notations and definitions concerning the schema matching problem. Section 3 is devoted to contributions (1) and (2). Section 4 provides the contributions (3) and (4). Section 5 concludes the paper. Due to space restrictions, some details had to be deferred to an appendix.

2 Preliminaries

We write $\mathbb{N}$ for the set of positive integers, $\mathbb{Q}$ for the set of rationals, and $\mathbb{Q}_{>0}$ for the set of positive rationals. For $m, n \in \mathbb{N}$ we let $[m, n] := \{i \in \mathbb{N} : m \leqslant i \leqslant n\}$ and $[n] := [1, n]$. A *rational-valued n-ary function on binary variables* (for short: *n-ary binary function*) is a function $f\colon \{0,1\}^n \to \mathbb{Q}$. A *partition* of a set S is a set P of non-empty, pairwise disjoint subsets of S such that $S = \bigcup_{X \in P} X$.

All graphs considered in this paper are finite undirected simple graphs. For a graph G we write $V(G)$ and $E(G)$ to denote the set of vertices and edges of G, respectively. In particular, every edge $e \in E(G)$ is a set consisting of exactly 2 elements of $V(G)$. A *weighted graph* is of the form (G, w) where G is a graph and $w : E(G) \to \mathbb{Q}_{>0}$ is a function that associates a positive *weight* $w(e)$ to every edge $e \in E(G)$. A *bipartite* graph is a graph G whose vertex set $V(G)$ is partitioned into two disjoint sets $V_1(G)$ and $V_2(G)$ such that every $e \in E(G)$ has one endpoint in $V_1(G)$ and one endpoint in $V_2(G)$. A bipartite graph G is *complete* if $E(G) = \{\{v_1, v_2\} : v_1 \in V_1(G),\ v_2 \in V_2(G)\}$.

Concerning database relations, we use the basic notation of [1] within the *named perspective*. In particular, we fix a countably infinite set **dom**, the set of potential entries in a database relation. A *relation schema* is a relation name R with an associated finite set $sort(R)$, the set of *attributes* of R. As usual, we will write $R[A_1, \ldots, A_r]$ to indicate that R is a relation schema with $sort(R) = \{A_1, \ldots, A_r\}$. An *$R$-tuple* is a mapping $t\colon sort(R) \to \mathbf{dom}$. An *$R$-relation* (or, a *database relation of schema $R[A_1, \ldots, A_r]$*) is a finite set of R-tuples.

Schema matching refers to the following data integration task. We are given two relation schemas $R_1[A_1, \ldots, A_{r_1}]$ and $R_2[B_1, \ldots, B_{r_2}]$, and potentially also an R_1-relation I_1 and an R_2-relation I_2. The ultimate goal is to find correspondences between attributes of R_1 and attributes of R_2 that indicate which (groups of) attributes of R_1 store similar information as certain (groups of) attributes of R_2. In this paper, we model such correspondences as a partition P of the set $W := W_1 \cup W_2$ for $W_1 := \{(1, A_1), \ldots, (1, A_{r_1})\}$ and $W_2 := \{(2, B_1), \ldots, (2, B_{r_2})\}$, such that every $X \in P$ is either a singleton or contains at least one element from each of the sets W_1 and W_2. The task of finding a suitable such partition also known as computing a *global matching* [10].

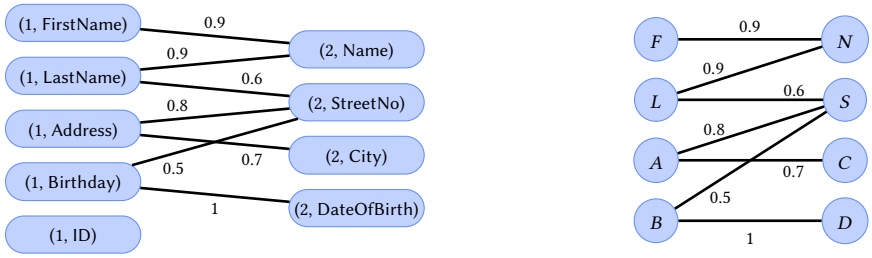
Let us consider the following example. Consider the relation schemas $R_1[\mathsf{FirstName}, \mathsf{LastName}, \mathsf{Address}, \mathsf{Birthday}, \mathsf{ID}]$ and $R_2[\mathsf{Name}, \mathsf{StreetNo}, \mathsf{City}, \mathsf{DateOfBirth}]$ and the associated relations I_1, I_2 depicted in Figure 1. Arguably, the most reasonable correspondences between the attributes are as follows. The attributes FirstName and LastName of R_1 store similar information as the attribute Name of R_2; this is modeled by the set $X_1 := \{(1, \mathsf{FirstName}), (1, \mathsf{LastName}), (2, \mathsf{Name})\}$. The attribute Address of R_1 stores similar information as the attributes StreetNo and City of R_2; this is modeled by the set $X_2 := \{(1, \mathsf{Address}), (2, \mathsf{StreetNo}), (2, \mathsf{City})\}$. The attribute Birthday of R_1 stores similar information as the attribute DateOfBirth of R_2; this is modeled by the set $X_3 := \{(1, \mathsf{Birthday}), (2, \mathsf{DateOfBirth})\}$. The attribute ID of R_1 has no analogue in attributes of R_2; this results in the singleton set $X_4 := \{(1, \mathsf{ID})\}$. All these correspondences are summarized in the partition $P := \{X_1, X_2, X_3, X_4\}$, which is a solution for the task of finding a global matching.

Practical tools (for an overview see Section 1) for finding suitable such schema matchings usually follow a two-stage process. The first stage is a *local schema matching*: Given two relation schemas $R_1[A_1, \ldots, A_{r_1}]$ and $R_2[B_1, \ldots, B_{r_2}]$, and potentially also an R_1-relation I_1 and an R_2-relation I_2, the goal is to assign suitable *similarity scores* to pairs of attributes of R_1 and R_2. These similarity scores serve as indicators on how similar the information stored in both attributes is to each other. We assume w.l.o.g. that similarity scores are normalized, i.e., they are non-negative rational values $\leqslant 1$. We model the result of this phase as a weighted bipartite graph (H, w) with $V_1(H) = W_1$, $V_2(H) = W_2$, and where $E(H)$ consists of exactly those edges $e = \{(1, A_i), (2, B_j)\}$ with $i \in [r_1]$ and $j \in [r_2]$ such that the attribute pair (A_i, B_j) has received a similarity score above a certain pre-defined threshold, and we let $w(e)$ be this similarity score. Figure 2a depicts an example of a bipartite weighted graph for our running example from Figure 1 at the end of the first stage.

In the second stage, the bipartite weighted graph (H, w) is processed with the goal to find a suitable global matching, i.e., a suitable partition P of $V(H)$ that is *valid* in the following sense: Every $X \in P$ is either a singleton or contains at least one element from each of the sets $V_1(H), V_2(H)$.

A *1:1 matching* for (H, w) is a suitable partition P of $V(H)$ where every $X \in P$ contains at most one element from each of the sets $V_1(H), V_2(H)$. A *1:N/N:1 matching* for (H, w) is a suitable partition P of $V(H)$ where for every $X \in P$ there exists an $i \in \{1, 2\}$ such that X contains exactly one element from $V_i(H)$ (and it may contain arbitrarily many elements from $V_{3-i}(H)$). A suitable valid partition P of $V(H)$ where each $X \in P$ contains arbitrarily many elements from both $V_1(H)$ and $V_2(H)$ is called an *N:M matching*.

In this paper we assume that the local schema matching has been taken care of, either by applying an automated matching method, or an interactive tool (see Section 1). This yields a bipartite weighted graph (H, w). The remainder of this paper focuses on the following task: Given



(a) Example (H, w). (b) Example (G, w).

Fig. 2. Example of a bipartite weighted graph (H, w) obtained at the end of the first stage, and the corresponding weighted graph (G, w) obtained from (H, w) by deleting all isolated nodes. For simplicity, the node labels of $V(G)$ are shortened.

(H, w), how to find a suitable global matching? To assign a precise meaning to this task, including a precise notion of which P are considered particularly suitable, the next section models the task as an optimization problem.

3 Modeling Global Matching as an Optimization Problem

In this section, we model the task of finding a suitable global matching as an optimization problem (Section 3.1), we provide an alternative characterization that is crucial for our proofs and algorithms (Section 3.2), and we prove that the problem is an NP-complete optimization problem (Section 3.3).

3.1 GMOP: Problem Definition

Recall from Section 2 that we are given a bipartite weighted graph (H, w) with $V_1(H) = \{(1, A_1), \ldots, (1, A_{r_1})\}$ and $V_2(H) = \{(2, B_1), \ldots, (2, B_{r_2})\}$. The aim of this subsection is to form a mathematically precise definition of what it means to find a "suitable" partition P of $V(H) = V_1(H) \cup V_2(H)$ so that every $X \in P$ represents a group of attributes in which similar information is stored.

Note that a node v of H is *isolated* (i.e., not contained in any edge of H) if, and only if, during the local matching phase no sufficiently similar attribute in the other relation schema was found for this node. Consequently, it is reasonable to demand that any *suitable* partition P

(A) should contain a singleton set $\{v\}$ for each isolated node of v of H.

We proceed by deleting all isolated nodes in H, and we write (G, w) to denote the resulting weighted bipartite graph (see Figure 2b for an example). Note that G has no isolated node and $0 < w(e) \leqslant 1$ for all $e \in E(G)$. Recall from Section 2 that $E(G)$ only contains edges whose weights exceed a certain pre-defined threshold. I.e., for any $i \in [2]$ and any node $v_i \in V_i(G)$ there exists at least one node $v_{3-i} \in V_{3-i}(G)$ such that the attributes represented by these nodes store somewhat similar information. It is therefore reasonable to demand that in any *suitable* partition P

(B) every $X \in P$ that is not of the form $\{v\}$ for an isolated node v of H, should contain at least one element from each of the sets $V_1(G)$ and $V_2(G)$.

Furthermore, as we want to ensure that any $X \in P$ represents a group of attributes that store mutually similar information, it is reasonable to demand that in any *suitable* partition P,

(C) the following should be satisfied by every $X \in P$:
for all $v_1 \in V_1(G)$ and $v_2 \in V_2(G)$ with $v_1, v_2 \in X$, we have $\{v_1, v_2\} \in E(G)$.

Definition 3.1. For any $X \subseteq V(G)$ let G_X be the subgraph of G induced by X, i.e., $V(G_X) = X$ and $E(G_X) := \{e \in E(G) \,:\, e \subseteq X\}$. For any set Q of pairwise disjoint subsets of $V(G)$, let G_Q be the subgraph of G with $V(G_Q) := \bigcup_{Y \in Q} Y$ and $E(G_Q) := \bigcup_{Y \in Q} E(G_Y)$.

Note that condition (C) states that G_X is a *complete* bipartite graph.

Definition 3.2. For any subgraph G' of G we let $w(G')$ be the *average weight* of edges in G', i.e., $w(G') = 0$ if $E(G') = \emptyset$ and $w(G') = \frac{1}{|E(G')|} \sum_{e \in E(G')} w(e)$ otherwise.

We want to ensure that the groups of attributes $X \in P$ are as small as possible while preserving high mutual similarity of the attributes. We use the average weight $w(G_X)$ as a measure for the mutual similarity of the attributes represented by X. We demand that in any *suitable* partition P,

(D) the following should be satisfied by every $X \in P$:
There does not exist a partition Q of X such that $|Q| \geqslant 2$ and
- every $Y \in Q$ is compatible with conditions (B), (C) in the sense that it contains at least one element from each of the sets $V_1(G)$ and $V_2(G)$, and G_Y is a complete bipartite graph, and
- $w(G_Q) \geqslant w(G_X)$.

Note that condition (D) states that a group $X \in P$ cannot be subdivided into smaller groups that still each meet the requirements (B) and (C) and have a cumulative average weight that is greater than or equal to the average weight for X.

This leads to the following formal definition. Recall that we assume that the preprocessing has already identified and deleted all attributes that cannot be matched adequately (see the paragraph following condition (A)). I.e., we henceforth consider the weighted graph (G, w) that is obtained from (H, w) by deleting all isolated nodes, and we demand that all the remaining attributes that are present as nodes in (G, w) must be matched with at least one other node from the opposite vertex set of G.

Definition 3.3 (Solution space $Sol(G, w)$)**.** Let (G, w) be a weighted bipartite graph that has no isolated node, and let $0 < w(e) \leqslant 1$ for all $e \in E(G)$. The *solution space* $Sol(G, w)$ of all *suitable partitions* consists of all partitions P of $V(G)$ that satisfy the following for every $X \in P$:

(1) G_X is a complete bipartite graph, and X contains at least one element from each of the sets $V_1(G)$ and $V_2(G)$, and
(2) there does not exist any partition Q of X such that $|Q| \geqslant 2$, $w(G_Q) \geqslant w(G_X)$, and G_Y is a complete bipartite graph and Y contains at least one element from each of the sets $V_1(G)$ and $V_2(G)$, for every $Y \in Q$.

The following lemma states that solutions always exist.

Lemma 3.4. *$Sol(G, w) \neq \emptyset$, for any weighted bipartite graph (G, w) that has no isolated node and where $0 < w(e) \leqslant 1$ for all $e \in E(G)$.*

The proof idea is as follows. We start with a maximum matching M of G in the graph-theoretic sense, i.e., M is a set of pairwise disjoint edges of G and $|M|$ is as large as possible. We say that a node v of G is *matched by* M iff $v \in \bigcup_{e \in M} e$. Since M is maximal, every node v of G that is *not* matched by M satisfies the following: all neighbors of v in G are matched by M. By assumption, G has no isolated node. Therefore, for each node v of G that is not matched by M, there exists a neighbor of v that *is* matched by M; we select one such neighbor u_v for each unmatched node v. Now, let G' be the graph with $V(G') = V(G)$, where $E(G')$ consists of all edges in M and the edges $\{v, u_v\}$ for all nodes v of G that are not matched by M. It can be shown that the sets of vertices of the connected components of G' form a partition P of $V(G)$ with $P \in Sol(G, w)$. See Appendix A for a detailed proof of Lemma 3.4.

Our proof of Lemma 3.4 also provides an algorithm for computing a particular $P \in Sol(G, w)$. But recall from Sections 1 and 2 that we would like to be able to find a $P \in Sol(G, w)$ that, in some precise sense, reflects the semantic correspondences between groups of attributes *as good as possible*. We measure the quality of a solution $P \in Sol(G, w)$ by its *average weight* $w(G_P)$ as defined in Definitions 3.1 and 3.2 (i.e., $V(G_P) = \bigcup_{X \in P} X = V(G)$ and $E(G_P) = \bigcup_{X \in P} E(G_X)$ and $w(G_P) = \frac{1}{|E(G_P)|} \sum_{e \in E(G_P)} w(e)$). The higher $w(G_P)$, the more suitable we reckon P.

Finally, this leads to the definition of the following optimization problem.

Definition 3.5 (The Global Matching Optimization Problem GMOP)**.**
The *global matching optimization problem* GMOP is the following maximization problem.

Input:	A weighted bipartite graph (G, w) that has no isolated node, and where $0 < w(e) \leqslant 1$ for every $e \in E(G)$.
Solution Space:	$Sol(G, w)$
Task:	Find a $P \in Sol(G, w)$ such that $w(G_P)$ is as large as possible.

We let

$$max(G, w) := \max\{w(G_P) : P \in Sol(G, w)\}, \quad \text{and}$$
$$Opt(G, w) := \{P \in Sol(G, w) : w(G_P) = max(G, w)\}.$$

Recall from the beginning of Section 3.1 that we start with a bipartite weighted graph (H, w) that represents the pairwise similarity values for the considered attributes; and these values are provided to us by a black-box matching method that has performed the *local schema matching* stage. Given such a (H, w), we consider the weighted graph (G, w), where G is obtained from H by deleting all isolated nodes. Now, any solution $P \in Sol(G, w)$ can be turned into a suitable solution P' for the *global matching* task on (H, w) by inserting into P the set $\{v\}$ for every isolated node v of H. We view those P' that are obtained from $P \in Opt(G, w)$ in this way as the *most suitable* solutions of the global matching task concerning (H, w).

In the remainder of this paper, we can therefore safely restrict attention to bipartite weighted graphs (G, w) that contain no isolated node and where $0 < w(e) \leqslant 1$ for every $e \in E(G)$.

3.2 GMOP: Characterization via 1:N / N:1 Matchings

This subsection's aim is to prove an alternative characterization of the solution space $Sol(G, w)$ of the problem GMOP.

Definition 3.6. Let G be a bipartite graph. A *1:N/N:1 matching* for G is a partition P of $V(G)$ such that for every $X \in P$ there exists an $i \in [2]$ such that X contains exactly one element from $V_i(G)$, and letting v denote this element, we have $\{v, u\} \in E(G)$ for every $u \in X$ with $u \neq v$.

Note that the connected components of the graph G_P associated with a 1:N / N:1 matching P for G somewhat look like a fan. We use the notion *fan-shaped* partition as a synonym for *1:N/N:1 matching*. The next main theorem characterizes $Sol(G, w)$ in terms of the 1:N / N:1 matchings for G.

Theorem 3.7. *Let (G, w) be a weighted bipartite graph that has no isolated node, and let $0 < w(e) \leqslant 1$ for all $e \in E(G)$. The solution space $Sol(G, w)$ consists of exactly those partitions P of $V(G)$ that are a 1:N/N:1 matching for G such that $|X| \geqslant 2$ for all $X \in P$.*

Proof. First, let P be a partition of $V(G)$ that is a 1:N / N:1 matching for G such that $|X| \geqslant 2$ for all $X \in P$. Note that, for every $X \in P$, each of the conditions (1) and (2) of Definition 3.3 is obviously satisfied, and hence $P \in Sol(G, w)$.

For the opposite direction, consider an arbitrary $P \in Sol(G, w)$. I.e., P is a partition of $V(G)$ such that every $X \in P$ satisfies the conditions (1) and (2) of Definition 3.3. In particular, $|X| \geqslant 2$ for all

$X \in P$. Assume for contradiction that P is not a 1:N / N:1 matching for G (cf. Definition 3.6). I.e., there exists an $X \in P$ such that for each $i \in [2]$ we have: $|X \cap V_i(G)| \neq 1$. For the remainder of this proof, let us consider a fixed such X.

Since $P \in \mathit{Sol}(G, w)$, we know that G_X is a complete bipartite graph and X contains a least one element from each of the sets $V_1(G)$, $V_2(G)$. Thus, $|X \cap V_i(G)| \geqslant 2$ for each $i \in [2]$.

For each $i \in [2]$ let $X_i := X \cap V_i(G)$. For $v_1 \in X_1$ and $v_2 \in X_2$ let $Q_{v_1,v_2} := \{Y_{v_1,v_2},\ Y'_{v_1,v_2}\}$ be the partition of X consisting of the sets $Y_{v_1,v_2} := \{v_1, v_2\}$ and $Y'_{v_1,v_2} := X \setminus \{v_1, v_2\}$.
The proofs of the following Claims 3.8 and 3.9 can be found in Appendix B.

Claim 3.8. For all $v_1 \in X_1$ and $v_2 \in X_2$ we have: $w(G_{Q_{v_1,v_2}}) < w(G_X)$.

For $i \in [2]$ let $n_i := |X_i| - 1$, and let $m := 1 + n_1 \cdot n_2$. Note that for all $v_1 \in X_1$ and $v_2 \in X_2$ we have:

$(*)$: $\quad m = |E(G_{Q_{v_1,v_2}})| \quad \text{and} \quad w(G_{Q_{v_1,v_2}}) = \frac{1}{m} \sum_{e \in E(G_{Q_{v_1,v_2}})} w(e)$.

Claim 3.9. $\sum_{\substack{v_1 \in X_1 \\ v_2 \in X_2}} \sum_{e \in E(G_{Q_{v_1,v_2}})} w(e) \quad = \quad m \cdot \sum_{e \in E(G_X)} w(e)$.

From Claim 3.9 and $(*)$ we obtain:

$$\sum_{\substack{v_1 \in X_1 \\ v_2 \in X_2}} w(G_{Q_{v_1,v_2}}) \quad = \quad \sum_{e \in E(G_X)} w(e) \quad = \quad |E(G_X)| \cdot w(G_X) \quad = \quad \sum_{\substack{v_1 \in X_1 \\ v_2 \in X_2}} w(G_X).$$

But this yields a contradiction, because from Claim 3.8 we obtain that $\sum_{\substack{v_1 \in X_1 \\ v_2 \in X_2}} w(G_{Q_{v_1,v_2}}) < \sum_{\substack{v_1 \in X_1 \\ v_2 \in X_2}} w(G_X)$.

This completes the proof of Theorem 3.7. □

Concerning the definition of the optimization problem GMOP (see Definition 3.5), one may wonder if one should further distinguish among the solutions in $\mathit{Opt}(G, w)$ by striving for only those $P \in \mathit{Opt}(G, w)$ for which $E(G_P)$ is as small as possible. From the following lemma we obtain that for the elements $P \in \mathit{Sol}(G, w)$ the number of edges in $E(G_P)$ differs only marginally (by a factor less than 2); see Appendix B for a proof.

Lemma 3.10. *Let (G, w) be a weighted bipartite graph that has no isolated node, and where $0 < w(e) \leqslant 1$ for every $e \in E(G)$. For $i \in [2]$ let $n_i := |V_i(G)|$.*
For any $P \in \mathit{Sol}(G, w)$ we have: $|E(G_P)| = n_1 + n_2 - |P|$ and $\max\{n_1, n_2\} \leqslant |E(G_P)| \leqslant n_1 + n_2 - 1$.

3.3 Complexity of GMOP

We now study the computational complexity of GMOP. In this subsection we assume that numbers in $\mathbb{N}$ are represented in *binary*, a number $q \in \mathbb{Q}_{>0}$ is represented by a pair $(n, d) \in \mathbb{N}^2$ with $q = \frac{n}{d}$, and the weight $w(e) \in \mathbb{Q}_{>0}$ of any edge e of a weighted graph (G, w) is represented by a pair (n_e, d_e) such that $d_e = d_{e'}$ for all $e, e' \in E(G)$. First, we note that GMOP is an *NP-optimization problem* [4].

Proposition 3.11. GMOP is an *NP-optimization problem*, i.e., it has the following properties:

(a) Upon input of a weighted graph (G, w) it can be decided in time polynomial in the size of the input whether (G, w) is a possible input for the problem GMOP (i.e., G is a bipartite graph that has no isolated node and where $0 < w(e) \leqslant 1$ for every $e \in E(G)$).
(b) Upon input of a weighted graph (G, w) that is a possible input for GMOP, and input of a set P consisting of subsets of $V(G)$, it can be decided in polynomial time whether $P \in \mathit{Sol}(G, w)$. Furthermore, every $P \in \mathit{Sol}(G, w)$ has size polynomial in the size of (G, w).
(c) Upon input of a weighted graph (G, w) that is a possible input for GMOP, and input of a $P \in \mathit{Sol}(G, w)$, the value $w(G_P)$ can be computed in polynomial time.

Proof. (a) and (c) are straightforward. The last statement of (b) is obvious. For proving statement (b), we use the characterization of $Sol(G, w)$ provided by Theorem 3.7: Upon input of a set P of subsets of $V(G)$ it can easily be checked in polynomial time if P is a 1:N / N:1 matching for G. □

Next, we will show that GMOP is an *NP-complete optimization problem*. In fact, we will show NP-hardness already for the following restricted version of GMOP.

Definition 3.12. 3GMOP is the restricted version of GMOP where inputs are inputs (G, w) of GMOP such that each node v of G has degree $\leqslant 3$ (i.e., v is contained in at most 3 edges of G).

D-3GMOP is the decision problem which receives as input an input (G, w) of 3GMOP and a number $q \in \mathbb{Q}_{>0}$, and the task is to decide whether there exists a $P \in Sol(G, w)$ with $w(G_P) \geqslant q$.

The remainder of this subsection is devoted to the proof of the following main result.

Theorem 3.13. 3GMOP *is an NP-complete optimization problem. I.e., it is an NP-optimization problem, and the decision problem* D-3GMOP *is NP-hard.*

Proof. Clearly, as GMOP is an NP-optimization problem, so is 3GMOP (because it can be checked in polynomial time whether each node of G has degree at most 3). For proving Theorem 3.13, it therefore suffices to prove that D-3GMOP is NP-hard.

Our proof proceeds by a polynomial-time reduction from Special-3SAT to D-3GMOP, where Special-3SAT is a suitably restricted version of the classical 3SAT-problem.

Recall that a 3SAT-*instance* is a finite non-empty set $\mathscr{C}$ of *clauses*; each *clause* $C \in \mathscr{C}$ is a non-empty set consisting of at most 3 *literals*, a *literal* is either of the form z (a so-called *positive literal*) or of the form $\neg z$ (a so-called *negative literal*), for some Boolean variable z. Let $Vars(\mathscr{C})$ be the set of Boolean variables that occur in $\mathscr{C}$.

3SAT is the decision problem which receives as input a 3SAT-instance $\mathscr{C}$, and the task is to decide whether there exists an *assignment* $\alpha\colon Vars(\mathscr{C}) \to \{0, 1\}$ that *satisfies* $\mathscr{C}$ in the sense that for every $C \in \mathscr{C}$ there exists a $z \in Vars(\mathscr{C})$ such that ($z \in C$ and $\alpha(z) = 1$) or ($\neg z \in C$ and $\alpha(z) = 0$).

Definition 3.14. Special-3SAT is the restricted version of 3SAT, where inputs are restricted to 3SAT-instances $\mathscr{C}$ that satisfy the following conditions:

(1) every $C \in \mathscr{C}$ has size $|C| \leqslant 3$ and contains $\leqslant 2$ positive literals and $\leqslant 2$ negative literals, and
(2) for each $z \in Vars(\mathscr{C})$, the positive literal z occurs in at most 2 clauses in $\mathscr{C}$, and the negative literal $\neg z$ occurs in at most 2 clauses in $\mathscr{C}$.

Lemma 3.15. *There is a polynomial-time reduction from* 3SAT *to* Special-3SAT.

The proof follows by introducing, for a given 3SAT-instance $\mathscr{C}$, for each $z \in Vars(\mathscr{C})$ new variables $z_0, \ldots, z_m$ and $\overline{z}_0, \ldots, \overline{z}_m$, for $m := |\mathscr{C}|$. We then add new clauses that ensure that all the variables $z_0, \ldots, z_m$ receive the same assignment (either 0 or 1) and that all the variables $\overline{z}_0, \ldots, \overline{z}_m$ receive the same, opposite assignment. Finally, for each $j \in [m]$ the j-th clause C_j in $\mathscr{C}$ is modified by suitably replacing each variable z that occurs in C_j with z_j or $\overline{z}_j$ in such a way that we obtain a Special-3SAT-instance. See Appendix C for a detailed proof of Lemma 3.15.

Lemma 3.16. *There is a polynomial-time reduction from* Special-3SAT *to* D-3GMOP.

Proof. Let $\mathscr{C}$ be a Special-3SAT-instance. We construct a D-3GMOP-instance $f(\mathscr{C}) = ((G, w), q)$ as follows. Let $m = |\mathscr{C}|$ and let $\mathscr{C} = \{C_1, \ldots, C_m\}$. Let $n = |Vars(\mathscr{C})|$ and let $Vars(\mathscr{C}) = \{z_1, \ldots, z_n\}$.

We introduce $2n$ nodes $v_{i,k}$ for $i \in [2]$ and $k \in [n]$, and further $4m$ nodes $c_{i,j}$, $d_{i,j}$ for $i \in [2]$ and $j \in [m]$. For $i \in [2]$ we let $V_i(G) := \{v_{i,k} : k \in [n]\} \cup \{c_{i,j},\ d_{i,j} : j \in [m]\}$. For $k \in [n]$, the nodes $v_{1,k}$ and $v_{2,k}$ will represent the literals z_k and $\neg z_k$, respectively. For $j \in [m]$, the nodes $c_{1,j}$ and $c_{2,j}$ are used to encode clause C_j; and the nodes $d_{1,j}$ and $d_{2,j}$ are called *fallback nodes* for clause C_j.

We let $E(G)$ consist of the following edges e, each associated with a weight $w(e)$:

- edge $e_{1,k,j} := \{v_{1,k}, c_{2,j}\}$ with $w(e_{1,k,j}) := 0.5$,
  for every $j \in [m]$ and every $k \in [n]$ such that C_j contains the positive literal z_k,
- edge $e_{2,k,j} := \{v_{2,k}, c_{1,j}\}$ with $w(e_{2,k,j}) := 0.5$,
  for every $j \in [m]$ and every $k \in [n]$ such that C_j contains the negative literal $\neg z_k$,
- edge $e_{z,k} := \{v_{1,k}, v_{2,k}\}$ with $w(e_{z,k}) := 1$, for every $k \in [n]$,
- edge $e_{d,j} := \{d_{1,j}, d_{2,j}\}$ with $w(e_{d,j}) := 1$, for every $j \in [m]$,
- edges $e_{1,cd,j} := \{c_{1,j}, d_{2,j}\}$ and $e_{2,cd,j} := \{c_{2,j}, d_{1,j}\}$ with $w(e_{1,cd,j}) := w(e_{2,cd,j}) := 0.5$,
  for every $j \in [m]$.

This completes the definition of (G, w). See Figure 3 for an illustration of the weighted graph (G, w) obtained for the Special-3SAT-instance $\mathscr{C}$ with $\mathscr{C} = \{C_1\}$ and $C_1 = \{z_1,\ z_2,\ \neg z_3\}$; edges of weight 1 are depicted by bold lines, edges of weight 0.5 by simple lines; nodes representing literals, clauses, and fallback nodes are colored blue, purple, and orange, respectively.

Clearly, (G, w) is a weighted bipartite graph that has no isolated node, and $0 < w(e) \leqslant 1$ for every $e \in E(G)$. Since $\mathscr{C}$ is a Special-3SAT-instance, it is straightforward to verify that each node of G has degree $\leqslant 3$. Hence, $f(\mathscr{C}) := \big((G, w), q\big)$ for

$$q \ := \ \tfrac{n+2m}{n+3m}$$

is an input for the decision problem D-3GMOP. Furthermore, upon input of $\mathscr{C}$, the tuple $f(\mathscr{C})$ can be computed in polynomial time.

To complete the proof of Lemma 3.16, it suffices to show that there exists an assignment $\alpha\colon \mathit{Vars}(\mathscr{C}) \to \{0,1\}$ that satisfies $\mathscr{C}$ if, and only if, there exists a $P \in \mathit{Sol}(G, w)$ with $w(G_P) \geqslant q$.

We achieve this by proving a series of claims, which use the following notation. The edges of G of weight 1 are called *heavy* edges, and the edges of weight 0.5 are called *light* edges. Note that $E(G)$ contains $n+m$ heavy edges. For each $P \in \mathit{Sol}(G, w)$ we write h_P and ℓ_P to denote the number of heavy edges and light edges, respectively, that belong to $E(G_P)$. See Appendix C for proofs of the following claims.

Claim 3.17. For all $P \in \mathit{Sol}(G, w)$: $\ w(G_P) \leqslant q$, and $w(G_P) = q \ \Leftrightarrow\ \big(\ h_P = n+m$ and $\ell_P = 2m\ \big)$.

Claim 3.18. If there exists a $P \in \mathit{Sol}(G, w)$ with $w(G_P) = q$, then there exists an assignment $\alpha\colon \mathit{Vars}(\mathscr{C}) \to \{0,1\}$ that satisfies $\mathscr{C}$.

Claim 3.19. If there exists an assignment $\alpha\colon \mathit{Vars}(\mathscr{C}) \to \{0,1\}$ that satisfies $\mathscr{C}$, then there exists a $P \in \mathit{Sol}(G, w)$ with $w(G_P) = q$.

From the Claims 3.17, 3.18, 3.19 we obtain that there exists an assignment $\alpha\colon \mathit{Vars}(\mathscr{C}) \to \{0,1\}$ that satisfies $\mathscr{C}$ if, and only if, there exists a $P \in \mathit{Sol}(G, w)$ with $w(G_P) \geqslant q$. This completes the proof of Lemma 3.16. □

It is well-known that 3SAT is NP-hard. Thus, by the Lemmas 3.15 and 3.16 we obtain that also D-3GMOP is NP-hard. This completes the proof of Theorem 3.13. □

4 Using Quantum Computing for Schema Matching

In this section, we provide some background on quantum computing and its relation to the problem of minimizing a rational-valued function on n binary variables (Section 4.1). We then translate the optimization problem GMOP into such a minimization problem (Section 4.2). Next, we delve deeper into the specifics of quantum computing (Section 4.3), which are crucial for understanding our experimental setup and the preliminary results (Section 4.4).

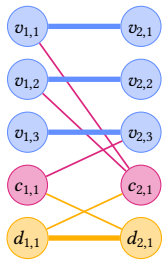


Fig. 3. Example of a weighted graph (G, w) constructed in the proof of Lemma 3.16.

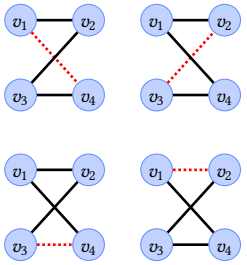


Fig. 4. Illustration concerning the definition of the function $g_{\overline{v}}$ in Section 4.2: all potential paths (edges depicted by solid lines) of length 3 on $v_1, v_3 \in V_1(G)$ and $v_2, v_4 \in V_2(G)$.

4.1 Some Foundations on Quantum Computing Relevant for this Paper

One variant of quantum computing, known as *adiabatic quantum annealing* [28], can address specific types of discrete optimization problems [15]. Broadly put, a system of partially interconnected wire loops (cooled down close to absolute zero and partly influenced by external magnetic forces) is constructed to represent a quadratic polynomial $p(\overline{x})$ on binary variables $\overline{x} = (x_1, \ldots, x_n)$. Thereby, the binary variables are represented by the orientations of magnetic fields induced by current flowing clockwise or counter-clockwise within the wire loops, and the energy landscape of this system represents the polynomial $p(\overline{x})$. The quantum state system will naturally thrive towards a lowest energy state. Attaining the minimal energy state, the measured orientations of magnetic fields represent an assignment to the binary variables that can be considered as a minimizer for $p(\overline{x})$. By means of tunneling effects (annealing) between local energy minima, a more efficient exploration of the solution space compared to classical algorithms can be achieved, potentially offering significant speedups for large-scale instances.

A well-known approach, called the *Quantum Approximate Optimization Algorithm (QAOA)*, transfers the concept of quantum annealing back to *circuit-based quantum computers* [12]. Quantum circuits can be described similarly to classical circuits as a sequence of operations (initialization, gates, measurements, etc.) on a set of quantum bits (qubits). Applying such a circuit is what is commonly referred to as quantum computing in the narrow sense; it can be implemented for example by means of quantum software and machines provided by IBM [17]. Circuit-based quantum computers are more similar to classical computers than quantum annealing devices, including their flexibility to arbitrarily combine operations. This gives quantum computers greater independence from a specific hardware structure and increases their practical scalability. The transformation from quantum annealing to circuit-based quantum computing is realized by discretizing the continuous annealing process and employing a hybrid quantum-classical system to efficiently explore the solution space. This alternative approach permits the consideration of more intricate objective functions and the incorporation of certain constraints, thereby extending its applicability to a wider range of applications.

QAOA is suitable for solving the following optimization problem: For a given function $f\colon \{0,1\}^n \to \mathbb{Q}$ (the *objective function*) and a certain domain $C \subseteq \{0,1\}^n$, find values $\overline{x} = (x_1, \ldots, x_n) \in C$ such that $f(x_1, \ldots, x_n)$ is as small as possible. I.e., the goal is to solve the following optimization problem: find an $\overline{x}$ in

$$\arg\min_{\overline{x} \in C} f(\overline{x}) \quad := \quad \{\, \overline{x} \in C \,:\, f(\overline{x}) \leqslant f(\overline{y}),\ \text{for all } \overline{y} \in C \,\}.$$

QAOA allows to consider arbitrary objective functions $f\colon \{0,1\}^n \to \mathbb{Q}$ and certain constrained domains $C \subseteq \{0,1\}^n$ [36]. In this paper, we will use the unconstrained setting, where $C = \{0,1\}^n$. A main complexity bottleneck is the number n of binary variables, which translates into the number of qubits needed to solve the problem using quantum computing. Additionally, the number and type of arithmetic operations of the function f can cause a computational overhead by increasing the required size and complexity of the quantum circuit.

4.2 Translating GMOP into a Minimization Problem Suitable for QAOA

The aim of this subsection is as follows. Given an input (G, w) for GMOP, i.e., (G, w) is a weighted bipartite graph that has no isolated node and where $0 < w(e) \leqslant 1$ for all $e \in E(G)$, we define a number $n := n_{(G,w)} \in \mathbb{N}$ and an n-ary binary function $f_{(G,w)}\colon \{0,1\}^n \to \mathbb{Q}$. We want $f_{(G,w)}$ to have the following properties:

(i) There is an injective mapping $\pi := \pi_{(G,w)}$ from the set $\{0,1\}^n$ to the set of all graphs G' with $V(G') = V(G)$ and $E(G') \subseteq E(G)$, such that for every $P \in \mathit{Sol}(G,w)$ we have $G_P \in \mathrm{img}(\pi)$.

(ii) There is a threshold $t_{\mathit{Sol}} \in \mathbb{Q}$ such that for all $\overline{x} \in \{0,1\}^n$ we have:
$f_{(G,w)}(\overline{x}) < t_{\mathit{Sol}} \iff \pi(\overline{x}) = G_P$ for some $P \in \mathit{Sol}(G,w)$.

(iii) For all $P, P' \in \mathit{Sol}(G,w)$ we have:
$w(G_P) \geqslant w(G_{P'}) \iff f_{(G,w)}(\overline{x}) \leqslant f_{(G,w)}(\overline{x}')$ for $\overline{x} := \pi^{-1}(G_P)$ and $\overline{x}' := \pi^{-1}(G_{P'})$.

Note that, in particular, (i)–(iii) imply:

(iv) For all $P \in \mathit{Sol}(G,w)$ we have: $P \in \mathit{Opt}(G,w) \iff \pi^{-1}(G_P) \in \arg\min_{\overline{x}\in\{0,1\}^n} f(\overline{x})$.

We start by choosing $n := n_{(G,w)} := |E(G)|$. Let $e_1, \ldots, e_n$ be a list of all edges in $E(G)$.
We define $\pi := \pi_{(G,w)}$ as follows. For every $\overline{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ let $\pi(\overline{x})$ be the graph G' with $V(G') = V(G)$ and $E(G') = \{e_i : i \in [n],\ x_i = 1\}$. Obviously, π is a bijective mapping from $\{0,1\}^n$ to the set of all subgraphs G' of G with $V(G') = V(G)$. Thus, property (i) is satisfied.

Before we can define $f_{(G,w)}$, we need to introduce some notations and intermediate functions. Alongside the definition of these functions, we will illustrate each step following the example given in Figure 2b. For every $v \in V(G)$ let $I_v := \{i \in [n] : v \in e_i\}$ and

$$f_v(\overline{x}) \ := \ \textstyle\prod_{i\in I_v}(1 - x_i), \quad \text{for all } \overline{x} = (x_1, \ldots, x_n) \in \{0,1\}^n.$$

We define $f_{\text{no-isol}}(\overline{x}) := \sum_{v\in V(G)} f_v(\overline{x})$, for all $\overline{x} \in \{0,1\}^n$. This definition obviously satisfies:

Fact 4.1. For all $\overline{x} \in \{0,1\}^n$, the following holds:
For all $v \in V(G)$, $f_v(\overline{x}) \in \{0,1\}$, and $f_v(\overline{x}) = 1 \iff v$ is an isolated node in the graph $\pi(\overline{x})$.
Furthermore, $f_{\text{no-isol}}(\overline{x}) \in \{0\} \cup \mathbb{N}$, and $f_{\text{no-isol}}(\overline{x}) = 0 \iff \pi(\overline{x})$ has no isolated node.

Example 4.2. Consider the weighted graph (G, w) depicted in Figure 2b. Note that G has eight nodes and seven edges, and $V(G) = \{F, L, A, B, N, S, C, D\}$. We let $n := 7$ and $e_1 := \{L, N\}$, $e_2 := \{L, S\}$, $e_3 := \{A, S\}$, $e_4 := \{B, S\}$, $e_5 := \{A, C\}$, $e_6 := \{B, D\}$, and $e_7 := \{F, N\}$. The order of the edges is chosen to fit to a later simplification step. When reading the outgoing edges of $V_1(G)$ in Figure 2b from top to bottom, the order is $e_7, e_1, e_2, e_3, e_5, e_4, e_6$.

The edges $e_1, \ldots, e_7$ are represented by the binary variables $x_1, \ldots, x_7$. For simplicity, we will write $x_{i(\{u,v\})}$ to refer to the particular x_i where $e_i = \{u, v\}$ for two adjacent nodes u and v.

We let $\overline{x} := (x_1, \ldots, x_7)$. For each $v \in V(G)$ the function $f_v(\overline{x})$ is defined as follows.

$$
\begin{aligned}
&f_F(\overline{x}) = 1-x_{i(\{F,N\})} = 1-x_7, \quad f_C(\overline{x}) = 1-x_{i(\{A,C\})} = 1-x_5, \quad f_D(\overline{x}) = 1-x_{i(\{B,D\})} = 1-x_6,\\
&f_L(\overline{x}) = (1 - x_{i(\{L,N\})}) \cdot (1 - x_{i(\{L,S\})}) = (1 - x_1) \cdot (1 - x_2),\\
&f_A(\overline{x}) = (1 - x_{i(\{A,S\})}) \cdot (1 - x_{i(\{A,C\})}) = (1 - x_3) \cdot (1 - x_5),\\
&f_B(\overline{x}) = (1 - x_{i(\{B,S\})}) \cdot (1 - x_{i(\{B,D\})}) = (1 - x_4) \cdot (1 - x_6),\\
&f_N(\overline{x}) = (1 - x_{i(\{F,N\})}) \cdot (1 - x_{i(\{L,N\})}) = (1 - x_7) \cdot (1 - x_1),\\
&f_S(\overline{x}) = (1 - x_{i(\{L,S\})}) \cdot (1 - x_{i(\{A,S\})}) \cdot (1 - x_{i(\{B,S\})}) = (1 - x_2) \cdot (1 - x_3) \cdot (1 - x_4).
\end{aligned}
$$

Summing up all these functions, we arrive at $f_{\text{no-isol}}(\overline{x}) := \sum_{v \in V(G)} f_v(\overline{x})$. ⌟

Our next goal is to define a function $f_{\text{fan-shaped}}$ with $f_{\text{fan-shaped}}(\overline{x}) \in \{0\} \cup \mathbb{N}$ and $f_{\text{fan-shaped}}(\overline{x}) = 0 \iff$ the graph $G' := \pi(\overline{x})$ is *fan-shaped* in the following sense: For every connected component X of G', there exists an $i \in [2]$ such that X contains exactly one element from $V_i(G)$ (and it may contain an arbitrary number of elements from $V_{3-i}(G)$). Note that G' is fan-shaped if, and only if, it does not contain a path of length 3 (i.e., a path consisting of 3 distinct edges).

To define $f_{\text{fan-shaped}}$, we need some more notation. For any $u, v \in V(G)$ let $i(\{u, v\}) \in \{0\} \cup [n]$ be defined as follows: if $\{u, v\} \notin E(G)$ then $i(\{u, v\}) := 0$; otherwise, $i(\{u, v\})$ is defined to be the particular $i \in [n]$ with $e_i = \{u, v\}$. Let $<$ be an arbitrary strict linear order on $V(G)$. Let W be the set of all tuples (v_1, v_2, v_3, v_4) of nodes of G with $v_1, v_3 \in V_1(G)$, $v_2, v_4 \in V_2(G)$, $v_1 < v_3$, and $v_2 < v_4$. For all $\overline{v} = (v_1, v_2, v_3, v_4) \in W$ and all $(x_0, x_1, \ldots, x_n) \in \{0, 1\}^{n+1}$ we let

$$
\begin{aligned}
g_{\overline{v}}(x_0, x_1, \ldots, x_n) \quad := \quad & x_{i(\{v_1,v_2\})} \cdot x_{i(\{v_2,v_3\})} \cdot x_{i(\{v_3,v_4\})} \;+\; x_{i(\{v_2,v_1\})} \cdot x_{i(\{v_1,v_4\})} \cdot x_{i(\{v_4,v_3\})} \;+\\
& x_{i(\{v_3,v_2\})} \cdot x_{i(\{v_2,v_1\})} \cdot x_{i(\{v_1,v_4\})} \;+\; x_{i(\{v_1,v_4\})} \cdot x_{i(\{v_4,v_3\})} \cdot x_{i(\{v_3,v_2\})},
\end{aligned}
$$

and we let $f_{\overline{v}}(x_1, \ldots, x_n) := g_{\overline{v}}(0, x_1, \ldots, x_n)$. See Figure 4 to gain some intuition for this definition. We let $f_{\text{fan-shaped}}(\overline{x}) := \sum_{\overline{v} \in W} f_{\overline{v}}(\overline{x})$, for all $\overline{x} \in \{0, 1\}^n$. It is not difficult to verify the following:

Fact 4.3. For all $\overline{x} \in \{0, 1\}^n$, the following holds: For all $\overline{v} \in W$ we have $f_{\overline{v}}(\overline{x}) \in \{0\} \cup \mathbb{N}$, and $f_{\overline{v}}(\overline{x}) \geqslant 1 \iff$ the graph $\pi(\overline{x})$ contains a path of length 3 on the nodes in $\overline{v}$.
Furthermore, $f_{\text{fan-shaped}}(\overline{x}) \in \{0\} \cup \mathbb{N}$, and $f_{\text{fan-shaped}}(\overline{x}) = 0 \iff \pi(\overline{x})$ is fan-shaped.

Example 4.4. Let us continue the construction from Example 4.2. Let $<$ be a strict linear order on $V(G)$ that increases from left to right and from top to bottom with respect to its representation in Figure 2b. For the weighted graph (G, w) depicted in Figure 2b and the tuple $\overline{v} := (F, N, L, S)$, the function $f_{\overline{v}}(\overline{x})$ is defined as follows:

$$
\begin{aligned}
& f_{(F,N,L,S)}(\overline{x})\\
= \;& x_{i(\{F,N\})} x_{i(\{L,N\})} x_{i(\{L,S\})} + x_{i(\{F,N\})} \cdot 0 \cdot x_{i(\{L,S\})} + x_{i(\{L,N\})} x_{i(\{F,N\})} \cdot 0 + 0 \cdot x_{i(\{L,S\})} x_{i(\{L,N\})}\\
= \;& x_{i(\{F,N\})} x_{i(\{L,N\})} x_{i(\{L,S\})} \quad = \quad x_7 \cdot x_1 \cdot x_2.
\end{aligned}
$$

The functions $f_{\overline{v}}(\overline{x})$ for all $\overline{v} \in W$ are defined analogously.

Overall, one obtains the function $f_{\text{fan-shaped}}(\overline{x}) = \sum_{\overline{v} \in W} f_{\overline{v}}(\overline{x})$. Note that $f_{\overline{v}}(\overline{x}) \neq 0$ only for those tuples $\overline{v} = (v_1, v_2, v_3, v_4) \in W$ where the induced subgraph of G on $\{v_1, v_2, v_3, v_4\}$ contains a path of length 3 (i.e., a path consisting of 3 edges). In our particular graph G, such a path exists if, and only if, $\{v_1, v_2, v_3, v_4\}$ is one of the sets $\{F, N, L, S\}$, $\{L, S, A, C\}$, $\{L, S, B, D\}$, $\{A, S, L, N\}$, $\{A, S, B, D\}$, $\{B, S, L, N\}$, $\{B, S, A, C\}$; and each of these sets yields one tuple $\overline{v}$ in W and an according function

$f_{\overline{v}}(\overline{x})$. In summary, we obtain that $f_{\text{fan-shaped}}(\overline{x}) =$

$$\begin{aligned} & x_{i(\{F,N\})}x_{i(\{L,N\})}x_{i(\{L,S\})} + x_{i(\{L,S\})}x_{i(\{A,S\})}x_{i(\{A,C\})} + x_{i(\{L,S\})}x_{i(\{B,S\})}x_{i(\{B,D\})} + \\ & x_{i(\{A,S\})}x_{i(\{L,S\})}x_{i(\{L,N\})} + x_{i(\{A,S\})}x_{i(\{B,S\})}x_{i(\{B,D\})} + x_{i(\{B,S\})}x_{i(\{L,S\})}x_{i(\{L,N\})} + \\ & x_{i(\{B,S\})}x_{i(\{A,S\})}x_{i(\{A,C\})} \\ = \; & x_7 \cdot x_1 \cdot x_2 + x_2 \cdot x_3 \cdot x_5 + x_2 \cdot x_4 \cdot x_6 + \\ & x_3 \cdot x_2 \cdot x_1 + x_3 \cdot x_4 \cdot x_6 + x_4 \cdot x_2 \cdot x_1 + x_4 \cdot x_3 \cdot x_5 \,. \end{aligned}$$

⌟

As a final ingredient, we define

$$f_{\text{avg}}(\overline{x}) \;:=\; -\,\frac{\sum_{i\in[n]} x_i \cdot w(e_i)}{\sum_{i\in[n]} x_i}, \quad \text{for all } \overline{x} = (x_1, \ldots, x_n) \in \{0,1\}^n \setminus \{0\}^n,$$

and $f_{\text{avg}}(\overline{0}) := 0$, where $\overline{0}$ denotes the n-tuple $(0, \ldots, 0)$. Obviously, we have:

Fact 4.5. For all $\overline{x} \in \{0,1\}^n$ and the graph $G' := \pi(\overline{x})$ we have: $f_{\text{avg}}(\overline{x}) = -w(G')$.
In particular, $f_{\text{avg}}(\overline{x}) \leqslant 0$.

Example 4.6. We continue the construction started in Examples 4.2 and 4.4. For $\overline{0} := (0,0,0,0,0,0,0)$ we have $f_{\text{avg}}(\overline{0}) = 0$, and for $\overline{x} \in \{0,1\}^7$ with $\overline{x} \neq \overline{0}$ we have $f_{\text{avg}}(\overline{x}) =$

$$\begin{aligned} & -\,\frac{0.9x_{i(\{F,N\})} + 0.9x_{i(\{L,N\})} + 0.6x_{i(\{L,S\})} + 0.8x_{i(\{A,S\})} + 0.7x_{i(\{A,C\})} + 0.5x_{i(\{B,S\})} + x_{i(\{B,D\})}}{x_{i(\{F,N\})} + x_{i(\{L,N\})} + x_{i(\{L,S\})} + x_{i(\{A,S\})} + x_{i(\{A,C\})} + x_{i(\{B,S\})} + x_{i(\{B,D\})}} \\ = \; & -\,\frac{0.9x_7 + 0.9x_1 + 0.6x_2 + 0.8x_3 + 0.7x_5 + 0.5x_4 + x_6}{x_7 + x_1 + x_2 + x_3 + x_5 + x_4 + x_6}. \end{aligned}$$

⌟

Finally, for all $\overline{x} \in \{0,1\}^n$ we let

$$f_{(G,w)}(\overline{x}) \;:=\; p_1 \cdot f_{\text{no-isol}}(\overline{x}) \;+\; p_2 \cdot f_{\text{fan-shaped}}(\overline{x}) \;+\; p_3 \cdot f_{\text{avg}}(\overline{x}),$$

where $p_1 := p_2 := p_3 := 1$ (these values are called *penalty values*; in future work we plan to perform an experimental evaluation on various different choices of these values).

Proposition 4.7. *For all inputs* (G, w) *of* GMOP, *the properties* (i)–(iii) *are satisfied for* $f_{(G,w)}$, $n := n_{(G,w)}$, $\pi := \pi_{(G,w)}$. *Concerning* (ii), *one can choose* $t_{Sol} := 0$.

Proof. This is an immediate consequence of the Facts 4.1–4.5. □

Simplification: Recall from Section 4.1 that a major efficiency bottleneck for using a QAOA algorithm to minimize a function f on n binary variables is the function's arity n. In the following, we modify $n = n_{(G,w)}$ and $f_{(G,w)}$ to a potentially smaller $\hat{n} := \hat{n}_{(G,w)}$ and an $\hat{n}$-ary function $\hat{f}_{(G,w)}$ that also have the properties (i)–(iii).

The *degree* $deg_G(v)$ of a node v in the graph G is the number of edges $e \in E(G)$ with $v \in e$. Let E_1 be the set of all those $e \in E(G)$ that contain a node v with $deg_G(v) = 1$. Let $E_2 := E(G) \setminus E_1$, and let

$$\hat{n} \;:=\; \hat{n}_{(G,w)} \;:=\; |E_2|\,.$$

Recall that at the beginning of Section 4.2 we fixed a list of edges $e_1, \ldots, e_n$ such that $E(G) = \{e_1, \ldots, e_n\}$. We assume w.l.o.g. that the first $\hat{n}$ edges in this list are the edges in E_2, i.e., $E_2 = \{e_1, \ldots, e_{\hat{n}}\}$. For any $\overline{x} = (x_1, \ldots, x_{\hat{n}}) \in \{0,1\}^{\hat{n}}$ we write $\overline{x1}$ for the n-tuple $(y_1, \ldots, y_n)$ with $y_i = 1$ for $i > \hat{n}$ and $y_i = x_i$ for $i \in [\hat{n}]$. We define $\hat{\pi} := \hat{\pi}_{(G,w)}$ and $\hat{f}_{(G,w)}$ via

$$\hat{\pi}(\overline{x}) \;:=\; \pi(\overline{x1}) \quad \text{and} \quad \hat{f}_{(G,w)}(\overline{x}) \;:=\; f_{(G,w)}(\overline{x1}), \quad \text{for all } \overline{x} \in \{0,1\}^{\hat{n}}.$$

Note that $\hat{\pi}(\overline{x})$ is the graph G' with $V(G') = V(G)$ and $E(G') = E_1 \cup \{e_i \,:\, i \in [\hat{n}],\ x_i = 1\}$.

Claim 4.8. Condition (i) is satisfied for $\hat{\pi}$ and $\hat{n}$.

Proof. Consider an arbitrary $P \in Sol(G, w)$. For each $i \in [\hat{n}]$ let $x_i := 1$ if $e_i \in E(G_P)$, and $x_i := 0$ otherwise. Let $\overline{x} = (x_1, \ldots, x_{\hat{n}})$. Clearly, $E(G_P) \subseteq E(\hat{\pi}(\overline{x})) \subseteq E(G_P) \cup E_1$. We are done by observing that $E_1 \subseteq E(G_P)$: Consider an arbitrary $e \in E_1$. Let $v \in e$ with $deg_G(v) = 1$. I.e., e is the only edge in $E(G)$ that contains v. Since P is a partition of $V(G)$, there exists an $X \in P$ with $v \in X$. According to Definition 3.3, G_X is a complete bipartite graph and X contains at least one element from each of the sets $V_1(G)$ and $V_2(G)$. Since e is the only edge in $E(G)$ with $v \in e$, we know that $e \subseteq X$. Thus, $e \in E(G_P)$. This concludes the proof of the claim. □

From Claim 4.8 and Proposition 4.7 we immediately obtain:

Corollary 4.9. *For all inputs* (G, w) *of* GMOP*, the properties* (i)–(iii) *are satisfied for* $\hat{f}_{(G,w)}$*,* $\hat{n} := \hat{n}_{(G,w)}$*,* $\hat{\pi} := \hat{\pi}_{(G,w)}$*. Concerning* (ii)*, one can choose* $t_{Sol} := 0$*.*

Example 4.10. We continue the construction started in the Examples 4.2, 4.4 and 4.6. Considering the simplified setting, the nodes F, C, D have degree 1, and these nodes are contained in the edges $e_5 = \{A, C\}$, $e_6 = \{B, D\}$ and $e_7 = \{F, N\}$.

Thus, in the simplified setting, each of the variables x_5, x_6, x_7 is replaced by the value 1, and we have $\hat{n} = 4$, and for all $\overline{x} = (x_1, x_2, x_3, x_4) \in \{0, 1\}^4$ we have:

$$\hat{f}_{\text{no-isol}}(\overline{x}) \;=\; (1 - x_1) \cdot (1 - x_2) \;+\; (1 - x_2) \cdot (1 - x_3) \cdot (1 - x_4)\,,$$

$$\begin{aligned}\hat{f}_{\text{fan-shaped}}(\overline{x}) \;&=\; x_1 \cdot x_2 \;+\; x_2 \cdot x_3 \;+\; x_2 \cdot x_4 \;+\\ &\quad\; x_3 \cdot x_2 \cdot x_1 \;+\; x_3 \cdot x_4 \;+\; x_4 \cdot x_2 \cdot x_1 \;+\; x_4 \cdot x_3\\ &=\; x_1 \cdot x_2 \;+\; x_2 \cdot x_3 \;+\; x_2 \cdot x_4 \;+\\ &\quad\; x_1 \cdot x_2 \cdot x_3 \;+\; 2 \cdot x_3 \cdot x_4 \;+\; x_1 \cdot x_2 \cdot x_4\,,\end{aligned}$$

$$\begin{aligned}\hat{f}_{\text{avg}}(\overline{x}) \;&=\; -\,\frac{0.9 + 0.9x_1 + 0.6x_2 + 0.8x_3 + 0.7 + 0.5x_4 + 1}{1 + x_1 + x_2 + x_3 + 1 + x_4 + 1}\\ &\quad\; -\,\frac{2.6 + 0.9x_1 + 0.6x_2 + 0.8x_3 + 0.5x_4}{3 + x_1 + x_2 + x_3 + x_4}\,.\end{aligned}$$

Finally, for the weighted graph (G, w) depicted in Figure 2b, we have

$$\hat{f}_{(G,w)}(\overline{x}) \;=\; \hat{f}_{\text{no-isol}}(\overline{x}) \;+\; \hat{f}_{\text{fan-shaped}}(\overline{x}) \;+\; \hat{f}_{\text{avg}}(\overline{x})\,.$$

⌟

In the subsequent parts of this paper, a tuple $\overline{x} \in \{0, 1\}^{\hat{n}}$ is called

- a *viable solution* if $\hat{f}_{(G,w)}(\overline{x}) < t_{Sol}$,
- an *optimal solution* if $\overline{x} \in \arg\min_{\overline{x} \in \{0,1\}^{\hat{n}}} \hat{f}_{(G,w)}(\overline{x})$.

According to the properties (i)–(iii), the *optimal* solutions $\overline{x}$ precisely correspond to the $P \in Opt(G, w)$, i.e., to the optimal answers for the problem GMOP on input (G, w); and the *viable* solutions precisely correspond to the partitions P of $V(G)$ that belong to $Sol(G, w)$, i.e., they can be viewed as "viable", albeit potentially suboptimal, answers for the problem GMOP on input (G, w).

4.3 Further Foundations Concerning QAOA Relevant for this Paper

Our next goal is to apply the quantum approximate optimization algorithm (QAOA) in order to find viable and, ideally, also optimal solutions for the problem of minimizing $\hat{f}_{(G,w)}(\overline{x})$. In fact, *all* our experiments succeeded and actually did find optimal solutions (cf. Section 4.4).

QAOA has a hybrid quantum-classical structure in which quantum and classical components alternate. During the quantum part, QAOA evaluates the expected value of the objective function with respect to a parameter-dependent quantum state. The classical output of the measurements on the quantum circuit acts as a feedback which is processed in the classical part of the algorithm to heuristically compute new parameters for the next quantum iteration. The new parameters are computed with the objective of minimizing the expected value of the objective function, thereby increasing the probability of (near-)optimal results, which eventually converges. Within each quantum circuit, the layers of quantum gates imitate the quantum annealing process. Each layer is defined by a function of the so-called *mixing Hamiltonian* which describes the initial state, and the *problem Hamiltonian* that describes the objective function. For simplicity, one can imagine that for advancing layers in the circuit, the quantum state shifts from the initial state (i.e., the ground state — which is the eigenvector associated with the minimum eigenvalue — of the *mixing Hamiltonian*) towards the solution state (i.e., the ground state of the *problem Hamiltonian*). In order to apply QAOA, several key parameters have to be set. These include the number of *layers*, which determines the discretization granularity of the simulated annealing process, the maximal number of *iterations* for an optimization process of the quantum circuit using classical hardware, as well as the number of *shots*. A shot is a measurement of the quantum state produced by the quantum circuit implemented for QAOA; it results in a tuple $\overline{x} \in \{0,1\}^{\hat{n}}$ that can be evaluated by $\hat{f}_{(G,w)}$.

Since quantum measurements are probabilistic, each tuple in $\{0,1\}^{\hat{n}}$ will have a certain probability of being measured. After running thousands of shots, we obtain a set of measurements yielding a *probability distribution*. It represents the likelihood of obtaining a certain tuple.

Note that not all measurements represent *viable solutions* in the sense defined at the end of Section 4.2. In a post-processing step, which is performed on classical hardware, all measured tuples must be checked. In the context of our work, we distinguish between non-viable tuples, solutions that are viable (but not necessarily optimal), and optimal solutions.

4.4 Proof-of-Concept: Implementation and Preliminary Experimental Results

As proof of concept, we compute schema matchings for a collection of benchmark scenarios, leveraging circuit-based quantum algorithms. Our goal is not to claim the superiority of quantum-based algorithms over classical approaches for schema matching. Such comparisons require careful calibration (e.g., tuning parameters such as layers, iterations, and shots) and consideration of the specific hardware characteristics (including gate availability, i.e. the set of quantum gates that a particular circuit-based quantum computer can implement, and the physical qubit topology).

Current quantum hardware is still quite limited in terms of the problem sizes it can address [21], and at the same time, simulators of quantum circuits can handle only a certain number of qubits with reasonable computational effort. Nevertheless, there are encouraging advances on the road to fault-tolerant quantum computing (cf., e.g., error correction experiments by Amazon [26], Google [2], Quantinuum [25], Rigetti [9]). Our primary objective with this proof-of-concept is to demonstrate that our formulation of the global matching problem is indeed solvable using QAOA. More generally, we also aim to illustrate that this line of theoretical research indeed holds potential for real-world applications, and warrants follow-up work from database theory and systems research.

Implementation: We implemented an end-to-end solution for schema matching using Qrisp [31] (version 0.5.4), a Python-based framework for programming circuit-based quantum algorithms.

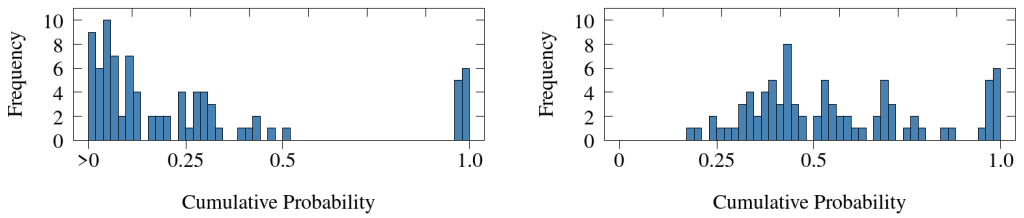
The first stage of local schema matching is executed entirely on classical hardware: the input scenario is parsed and all pairs of attributes are scored using the Cupid matching method [22] (in the implementation from Valentine [20]). We are not restricted to Cupid, and it is straightforward to exchange the matching method, e.g. for another one provided by the Valentine framework.

The global matching problem is then encoded for evaluation within QAOA, as described in Section 4.2. The QAOA algorithm can be evaluated on quantum hardware or also on a quantum simulator. Postprocessing, i.e. checking the individual solutions, and translating the bit vectors back to a format that is human-consumable, is again performed on classical hardware.

Execution environment: We run our experiments on a server with two 20-core Intel Xeon Gold 6242R 3,1 GHz processors and 192 GB of RAM, running Arch Linux. For QAOA evaluation, we use a CPU-based state vector simulator of a circuit-based quantum computer, which comes with Qrisp. This high-performance simulator utilizes sparse matrices to store and process quantum states.

Schema matching scenarios: We consider scenarios from the ScheMatch collection [34]. Each scenario consists of pairs of relations from various domains (e.g., bibliographic or shipping data). Among the original 291 scenarios, we excluded approx. 30% as trivial to solve (where all edges contain a node of degree 1), as well as scenarios where the Cupid matching method exceeded a 5-minute timeout. Moreover, we excluded scenarios whose encoding requires more than 10 quantum variables (a data type in Qrisp that we use to represent the binary variables). This is not a limitation of the simulator software, but a practicable limitation to avoid excessively long simulation runs. Among the remaining 85 scenarios, two are known to go beyond 1:1 matchings.

Experiment design: We perform schema matching on the subset of ScheMatch scenarios, using the implementation described above. We configure QAOA with 3 layers, 100 iterations, and 5K shots per scenario. For a fixed scenario (GMOP instance) run with 5K shots, we sum the probabilities of all optimal and viable solutions found. Across the entire collection of scenarios, we then compute the frequencies of these cumulative optimal/viable probabilities. Figure 5 shows the histograms.



(a) Optimal solutions. (b) Viable solutions.

Fig. 5. Frequencies of cumulative probabilities for ScheMatch scenarios. All cumulative probabilities non-zero.

Results: In Figure 5a, we show the cumulative probabilities of finding an optimal solution among 5K shots, across all considered scenarios. In 11 of 85 scenarios, nearly all candidate solutions computed were optimal (with cumulative probabilities in high nineties or even 100%). For most scenarios, the probability of finding an optimal solution is below 50%, yet never zero (indicated by axis label “>0”). This means that at least one optimal solution was found for each scenario, so we consistently solved the schema matching problem. For 87% of the scenarios, we found a single optimal solution (but we may have found the same solution several times). For the remaining scenarios, we found two or even three alternative optimal solutions to the global matching problem.

When we consider the frequencies of cumulative probabilities for viable solutions, shown in Figure 5b, the median cumulative probability improves noticeably, as expected.

Discussion: Already in this first batch of experiments, we achieve very promising results: Even without custom tuning, we found at least one optimal solution per scenario.

In scenarios where more than one optimal solution to the global matching problem was found, it is up to the domain expert to choose. Although benchmarks may provide a "ground truth", related work on schema matching reports on a user study where almost no two users could agree on the intended match result for a given matching task [23]. Consequently, automatically computed solutions for schema matching will always have to be inspected by domain experts, and possibly even manually adapted. These experts might actually be interested in inspecting the non-optimal but viable solutions when they explore the solution space, to obtain a wider pool of solution candidates for their data integration problem.

Given that the experiment is probabilistic, repeating the experiment will not produce the exact same distributions. However, by increasing the number of shots in QAOA, we can approximate the outcome more robustly. Overall, we can conclude that these promising results motivate further research in this area.

5 Final Remarks

We introduced the Global Matching Optimization Problem (GMOP) as a formal definition in order to find a suitable global matching for two relation schemas. We characterized the solutions of this task as 1:N / N:1 matchings. We showed that GMOP is an NP-complete optimization problem, even when inputs are restricted to be of degree at most 3 (i.e., for any attribute of one relation schema there are at most 3 potentially matching attributes of the other relation schema). We reduced GMOP to the problem of minimizing a particular rational-valued function on binary variables. This enabled us to use the Quantum Approximate Optimization Algorithm (QAOA) to solve GMOP.

In a micro-experiment, we applied an implementation of our approach to a collection of scenarios from a schema matching benchmark. Our initial experiments highlight the potential of using QAOA on quantum hardware to solve the schema matching problem. Even without fine-tuning parameters, we have successfully computed optimal solutions. This confirms the practicality of our approach.

Looking ahead, we plan to expand these experiments towards systematic evaluations that uncover both the strengths and limitations of our approach. A further task is to consider suitable variants of GMOP that also allow for solutions that are N:M matchings. A promising candidate is obtained by a modified version of Definition 3.3, where in item (2) the condition $w(G_Q) \geqslant w(G_X)$ is replaced by the condition $\min_{Y \in Q} w(G_Y) \geqslant w(G_X)$. We plan to investigate this in future work.

We see research at the intersection of databases and quantum computing as a unique opportunity to provide feedback for the quantum computing community in terms of which functionality and software tools would be required to utilize quantum computing for tasks related to databases. We believe it is important to study use cases already today, and to develop the algorithms and tools to utilize potential quantum advantages as the technology matures and quantum computing hardware is improved. That way, the database community has the potential to influence the early development of quantum software engineering libraries (in fact, we have already provided feedback to the Qrisp team). At the same time, this can help database scientists to gain a better understanding of the capabilities of quantum computing.

Acknowledgments

The authors thank Edgar Yépez (Universität Passau) for providing the similarity matrices for the ScheMatch scenarios, computed with the Valentine framework for our experiments.
Luisa Gerlach was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 414984028 – CRC 1404 FONDA (https://fonda.hu-berlin.de).

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley. http://webdam.inria.fr/Alice/

[2] Rajeev Acharya, Dmitry A. Abanin, Laleh Aghababaie-Beni, Igor Aleiner, et al. 2025. Quantum error correction below the surface code threshold. *Nature* 638 (Feb. 2025), 920–926. https://doi.org/10.1038/s41586-024-08449-y

[3] Altova. 2025. Altova MapForce. https://www.altova.com/de/mapforce Last accessed: 2025-04-01.

[4] G. Ausiello, P. Crescenzi, and M. Protasi. 1995. Approximate solution of NP optimization problems. *Theoretical Computer Science* 150, 1 (1995), 1–55. https://doi.org/10.1016/0304-3975(94)00291-P

[5] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm (Eds.). 2011. *Schema Matching and Mapping*. Springer. https://doi.org/10.1007/978-3-642-16518-4

[6] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. 2011. Generic Schema Matching, Ten Years Later. *Proc. VLDB Endow.* 4, 11 (2011), 695–701. http://www.vldb.org/pvldb/vol4/p695-bernstein_madhavan_rahm.pdf

[7] Lukas Bischof, Stefan Teodoropol, Rudolf M. Füchslin, and Kurt Stockinger. 2025. Hybrid quantum neural networks show strongly reduced need for free parameters in entity matching. *Scientific Reports* 15, 1 (Feb. 2025). https://doi.org/10.1038/s41598-025-88177-z

[8] Tim Bittner and Sven Groppe. 2020. Avoiding blocking by scheduling transactions using quantum annealing. In *Proc. 24th International Database Engineering & Applications Symposium*. 21:1–21:10. https://doi.org/10.1145/3410566.3410593

[9] Laura Caune, Luka Skoric, Nick S. Blunt, Archibald Ruban, et al. 2024. Demonstrating real-time and low-latency quantum error correction with superconducting qubits. arXiv:2410.05202 [quant-ph] https://arxiv.org/abs/2410.05202

[10] Jérôme Euzenat and Pavel Shvaiko. 2013. *Global Matching Methods*. Springer Berlin Heidelberg, 121–148. https://doi.org/10.1007/978-3-642-38721-0_6

[11] Tobias Fankhauser, Marc E. Solèr, Rudolf Marcel Füchslin, and Kurt Stockinger. 2023. Multiple Query Optimization Using a Gate-Based Quantum Computer. *IEEE Access* 11 (2023), 114031–114043. https://doi.org/10.1109/ACCESS.2023.3324253

[12] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph] http://arxiv.org/abs/1411.4028

[13] Kristin Fritsch and Stefanie Scherzinger. 2023. Solving Hard Variants of Database Schema Matching on Quantum Computers. *Proc. VLDB Endow.* 16, 12 (2023), 3990–3993. https://doi.org/10.14778/3611540.3611603

[14] Avigdor Gal. 2005. On the Cardinality of Schema Matching. In *Proc. On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*. 947–956. https://doi.org/10.1007/11575863_116

[15] Fred Glover, Gary Kochenberger, and Yu Du. 2019. A Tutorial on Formulating and Using QUBO Models. arXiv:1811.11538 [quant-ph] http://arxiv.org/abs/1811.11538

[16] Laura M. Haas, Mauricio A. Hernández, Howard Ho, Lucian Popa, and Mary Roth. 2005. Clio grows up: from research prototype to industrial tool. In *Proc. of the ACM SIGMOD International Conference on Management of Data*. 805–810. https://doi.org/10.1145/1066157.1066252

[17] IBM. 2025. IBM Quantum. https://quantum.ibm.com/?src=ibm-quantum-homepage Accessed: 2025-04-01.

[18] Anastasios Kementsietsidis. 2009. Schema Matching. In *Encyclopedia of Database Systems*, Ling Liu and M. Tamer Özsu (Eds.). Springer US, Boston, MA, 2494–2497. https://doi.org/10.1007/978-0-387-39940-9_962

[19] Florian Kittelmann, Pavel Sulimov, and Kurt Stockinger. 2024. QardEst: Using Quantum Machine Learning for Cardinality Estimation of Join Queries. In *Proc. Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications*. https://doi.org/10.1145/3665225.3665444

[20] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, et al. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *Proc. 37th IEEE International Conference on Data Engineering*. 468–479. https://doi.org/10.1109/ICDE51399.2021.00047

[21] Frank Leymann and Johanna Barzen. 2020. The bitter truth about gate-based quantum algorithms in the NISQ era. *Quantum Science and Technology* 5, 4 (Sept. 2020), 044007. https://doi.org/10.1088/2058-9565/abae7d

[22] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. 2001. Generic Schema Matching with Cupid. In *Proc. 27th International Conference on Very Large Data Bases*. 49–58. http://www.vldb.org/conf/2001/P049.pdf

[23] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. 2002. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *Proc. 18th International Conference on Data Engineering*. 117–128. https://doi.org/10.1109/ICDE.2002.994702

[24] Renée J. Miller, Mauricio A. Hernández, Laura M. Haas, Lingling Yan, C. T. Howard Ho, Ronald Fagin, and Lucian Popa. 2001. The Clio project: managing heterogeneity. *SIGMOD Rec.* 30, 1 (March 2001), 78–83. https://doi.org/10.1145/373626.373713

[25] A. Paetznick, M. P. da Silva, C. Ryan-Anderson, J. M. Bello-Rivas, et al. 2024. Demonstration of logical qubits and repeated error correction with better-than-physical error rates. arXiv:2404.02280 [quant-ph] https://arxiv.org/abs/2404.02280

[26] Harald Putterman, Kyungjoo Noh, Connor T. Hann, Gregory S. MacCabe, et al. 2025. Hardware-efficient quantum error correction via concatenated bosonic qubits. *Nature* 638 (Feb. 2025), 927–934. https://doi.org/10.1038/s41586-025-

08642-7

[27] Erhard Rahm and Philip A. Bernstein. 2001. A survey of approaches to automatic schema matching. *VLDB J.* 10, 4 (2001), 334–350. https://doi.org/10.1007/S007780100057

[28] Atanu Rajak, Sei Suzuki, Amit Dutta, and Bikas K. Chakrabarti. 2023. Quantum Annealing: An Overview. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 381, 2241 (2023), 20210417. https://doi.org/10.1098/rsta.2021.0417

[29] Christoph Roch, David Winderl, Claudia Linnhoff-Popien, and Sebastian Feld. 2022. A Quantum Annealing Approach for Solving Hard Variants of the Stable Marriage Problem. In *Proc. Innovations for Community Services - 22nd International Conference*. 294–307. https://doi.org/10.1007/978-3-031-06668-9_21

[30] Manuel Schönberger, Stefanie Scherzinger, and Wolfgang Mauerer. 2023. Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. *Proc. ACM Manag. Data* 1, 1 (2023), 92:1–92:27. https://doi.org/10.1145/3588946

[31] Raphael Seidel, Sebastian Bock, René Zander, Matic Petric, Niklas Steinmann, Nikolay Tcholtchev, and Manfred Hauswirth. 2024. Qrisp: A Framework for Compilable High-Level Programming of Gate-Based Quantum Computers. *CoRR* abs/2406.14792 (2024). https://doi.org/10.48550/ARXIV.2406.14792

[32] Immanuel Trummer and Christoph Koch. 2016. Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer. *Proc. VLDB Endow.* 9, 9 (2016), 648–659. https://doi.org/10.14778/2947618.2947621

[33] Immanuel Trummer and Davide Venturelli. 2024. Leveraging Quantum Computing for Database Index Selection. In *Proc. Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications*. https://doi.org/10.1145/3665225.3665445

[34] Alexander Vielhauer. 2024. ScheMatch. Big Data Analytics Group, Thorsten Papenbrock, Marburg University. https://github.com/avielhauer/schematch Accessed: 2024-12-08.

[35] Tobias Winker, Umut Çalikyilmaz, Le Gruenwald, and Sven Groppe. 2023. Quantum Machine Learning for Join Order Optimization using Variational Quantum Circuits. In *Proc. International Workshop on Big Data in Emergent Distributed Environments*. 5:1–5:7. https://doi.org/10.1145/3579142.3594299

[36] Rene Zander, Raphael Seidel, Matteo Inajetovic, Niklas Steinmann, and Matic Petric. 2024. Solving the Product Breakdown Structure Problem with constrained QAOA . In *Proc. IEEE International Conference on Quantum Computing and Engineering*. 462–463. https://doi.org/10.1109/QCE60285.2024.10356

APPENDIX

A Proof Omitted in Section 3.1

Proof of Lemma 3.4.
Let $V := V_1(G) \cup V_2(G)$. For any set P consisting of pairwise disjoint subsets of V, we say that a node $v \in V$ is *matched by* P if $v \in X$ for some $X \in P$.

Let M be a maximum matching of G in the graph-theoretic sense, i.e., M is a set of pairwise disjoint edges of G and $|M|$ is as large as possible.

Consider an arbitrary node $v \in V$ that is not matched by M, and let v' be an arbitrary neighbor of v in G (such v' exists because, by assumption, G has no isolated node). Since M is maximal, v' must be matched by M, because otherwise $M \cup \{\{v, v'\}\}$ would be a strictly larger set of pairwise disjoint edges of G. For later use we summarize this as

Fact 1: For each $v \in V$ that is not matched by M, all neighbors v' of v in G are matched by M.

Let $m := |M|$ and let $\{e_1, \ldots, e_m\} = M$. We start with $P_0 := M$, and we subsequently define sets $P_1, \ldots, P_m$ such that, for all $j \in [m]$, P_j consists of pairwise disjoint subsets of V, as follows:

Assume that, for some $j \in [m]$, the set P_{j-1} has already been defined. Let $u_1, u_2 \in V$ such that $e_j = \{u_1, u_2\}$. For $i \in [2]$ let N_i be the set of all neighbors of u_i in G that are not matched by P_{j-1}.

- If $N_1 = N_2 = \emptyset$ then let $P_j := P_{j-1}$.
- Otherwise, if for some $i \in [2]$ we have $N_i \neq \emptyset$ and $N_{3-i} = \emptyset$ then let P_j be obtained from P_{j-1} by replacing the set e_j with the set $e_j \cup N_i$, i.e., $P_j = (P_{j-1} \setminus \{e_j\}) \cup \{e_j \cup N_i\}$.
- Otherwise, $N_1 \neq \emptyset$ and $N_2 \neq \emptyset$, i.e., there exists $w_i \in N_i$ for each $i \in [2]$. But then, $M' := (M \setminus \{\{u_1, u_2\}\}) \cup \{\{u_1, w_1\}, \{u_2, w_2\}\}$ is a set of pairwise disjoint edges of G strictly larger than M. This contradicts our choice of M. Hence, the case $N_1 \neq \emptyset$ and $N_2 \neq \emptyset$ cannot occur.

It is straightforward to verify the following: (1) All nodes that are matched by M are also matched by P_j, (2) the elements of P_j are pairwise disjoint and each of size $\geqslant 2$ and each contains at least one node from each of the sets $V_1(G)$ and $V_2(G)$, (3) all nodes that are neighbors in G of some node that belongs to $\bigcup_{k=1}^{j} e_k$ are matched by P_j, and (4) for every $X \in P_j$, the induced subgraph G_X is a complete bipartite graph, where there exists a node in X that is adjacent to all other nodes in X. For $j = m$, we additionally obtain from *Fact 1* and the assumption that G has no isolated node that P_m is a partition of V.

It is straightforward to verify that $P_m \in Sol(G, w)$ (note that statement (4) implies that, in particular, condition (2) of Definition 3.3 is satisfied). This completes the proof of Lemma 3.4. □

B Proofs Omitted in Section 3.2

Proof of Claim 3.8.
Note that $|Q_{v_1,v_2}| \geqslant 2$, and for every $Y \in Q_{v_1,v_2}$ the following is true: G_Y is a complete bipartite graph and Y contains at least one element from each of the sets $V_1(G)$, $V_2(G)$. Since $P \in Sol(G, w)$, we obtain from item (2) of Definition 3.3 that $w(G_{Q_{v_1,v_2}}) < w(G_X)$.
This completes the proof of Claim 3.8. □

Proof of Claim 3.9.
Let $v_1 \in X_1$ and $v_2 \in X_2$. Let $e_{v_1,v_2} := \{v_1, v_2\}$, and for $i \in [2]$ let $E_{v_i} := \{e \in E(G_X) : v_i \in e\}$. Note that $E(G_X)$ is the union of the disjoint sets $E(G_{Q_{v_1,v_2}})$ and $E_{v_1} \setminus \{e_{v_1,v_2}\}$ and $E_{v_2} \setminus \{e_{v_1,v_2}\}$. Thus,

$$\sum_{e \in E(G_X)} w(e) \quad = \quad \sum_{e \in E(G_{Q_{v_1,v_2}})} w(e) \quad + \quad \sum_{e \in E_{v_1}} w(e) \quad + \quad \sum_{e \in E_{v_2}} w(e) \quad - \quad 2 \cdot w(e_{v_1,v_2}).$$

I.e., letting $W_X := \sum_{e \in E(G_X)} w(e)$ and $W_{v_i} := \sum_{e \in E_{v_i}} w(e)$ for $i \in [2]$, we obtain

$$\sum_{e \in E(G_{Q_{v_1,v_2}})} w(e) \quad = \quad W_X \quad - \quad W_{v_1} \quad - \quad W_{v_2} \quad + \quad 2 \cdot w(e_{v_1,v_2}).$$

Consequently, letting $n := |X_1| \cdot |X_2|$ we obtain:

$$\sum_{\substack{v_1 \in X_1 \\ v_2 \in X_2}} \sum_{e \in E(G_{Q_{v_1,v_2}})} w(e) \quad = \quad n \cdot W_X \quad - \quad |X_2| \cdot \sum_{v_1 \in X_1} W_{v_1} \quad - \quad |X_1| \cdot \sum_{v_2 \in X_2} W_{v_2} \quad + \quad 2 \cdot W_X.$$

Furthermore, note that $\bigcup_{v_1 \in X_1} E_{v_1} = E(G_X) = \bigcup_{v_2 \in X_2} E_{v_2}$. Hence, $\sum_{v_1 \in X_1} W_{v_1} = W_X = \sum_{v_2 \in X_2} W_{v_2}$.
In summary, we obtain:

$$\sum_{\substack{v_1 \in X_1 \\ v_2 \in X_2}} \sum_{e \in E(G_{Q_{v_1,v_2}})} w(e) \quad = \quad n \cdot W_X \quad - \quad |X_2| \cdot W_X \quad - \quad |X_1| \cdot W_X \quad + \quad 2 \cdot W_X \quad = \quad (n - n_1 - n_2) \cdot W_X.$$

We are done by noting that $n - n_1 - n_2 = (n_1{+}1) \cdot (n_2{+}1) - n_1 - n_2 = n_1 \cdot n_2 + 1 = m$.
This completes the proof of Claim 3.9. □

Proof of Lemma 3.10.
Let $P \in Sol(G, w)$. From Theorem 3.7 we know that P is a 1:N / N:1 matching for G such that $|X| \geqslant 2$ for all $X \in P$. Thus, $|E(G_P)| = \sum_{X \in P} |E(G_X)| = \sum_{X \in P} (|X| - 1) = \left(\sum_{X \in P} |X| \right) - |P|$. Since P is a partition of $V(G)$, we have: $\sum_{X \in P} |X| = n_1 + n_2$. Hence, $|E(G_P)| = n_1 + n_2 - |P|$.

Note that $|P| \geqslant 1$, and hence $|E(G_P)| \leqslant n_1 + n_2 - 1$. On the other hand, $|E(G_P)| \geqslant \max\{n_1, n_2\}$ due to the following reasoning: $E(G_P) = \bigcup_{X \in P} E(G_X)$. Let $i \in [2]$. For every $v \in V_i(G)$ there exists an $X \in P$ such that $v \in X$. According to Definition 3.3, G_X is a complete bipartite graph and X contains at least one element from each of the sets $V_1(G)$ and $V_2(G)$. Thus, X contains a node $u_v \in V_{3-i}(G)$ such that $\{v, u_v\} \in E(G_X)$. Thus $F_i := \{\{v, u_v\} : v \in V_i(G)\}$ is a set of edges in G_P of size $|F_i| = |V_i(G)| = n_i$. Hence, $|E(G_P)| \geqslant n_i$ for each $i \in [2]$. This completes the proof of Lemma 3.10. □

C Proofs Omitted in Section 3.3

Proof of Lemma 3.15.
Let $\mathscr{C}$ be a 3SAT-instance. We construct a Special-3SAT-instance $f(\mathscr{C})$ as follows.
Let $m = |\mathscr{C}|$ and let $\mathscr{C} = \{C_1, \ldots, C_m\}$. Let $n = |Vars(\mathscr{C})|$ and let $Vars(\mathscr{C}) = \{z_1, \ldots, z_n\}$.
Introduce $2n \cdot (m{+}1)$ new Boolean variables $z_{k,j}$ and $\overline{z}_{k,j}$ for $k \in [n]$ and $j \in \{0\} \cup [m]$.
For all $k \in [n]$, define the following clauses:

$$\begin{aligned} D_{k,0} &:= \{z_{k,0}, \neg z_{k,m}\} \quad \text{and} \quad D_{k,j} := \{z_{k,j}, \neg z_{k,j-1}\}, \quad \text{for all } j \in [m]. \\ \overline{D}_{k,0} &:= \{\overline{z}_{k,0}, \neg \overline{z}_{k,m}\} \quad \text{and} \quad \overline{D}_{k,j} := \{\overline{z}_{k,j}, \neg \overline{z}_{k,j-1}\}, \quad \text{for all } j \in [m]. \\ E_{k,+} &:= \{z_{k,0}, \overline{z}_{k,0}\} \quad \text{and} \quad E_{k,-} := \{\neg z_{k,0}, \neg \overline{z}_{k,0}\}. \end{aligned}$$

Let $\mathscr{K}$ be the set of all these clauses, i.e., $\mathscr{K}$ consists of the clauses $E_{k,+}$, $E_{k,-}$, $D_{k,j}$, $\overline{D}_{k,j}$ for all $k \in [n]$ and $j \in \{0\} \cup [m]$. Note that an assignment $\beta : Vars(\mathscr{K}) \to \{0, 1\}$ satisfies $\mathscr{K}$ if, and only if, for all $k \in [n]$ we have: $\beta(z_{k,0}) = 1 \iff \beta(\overline{z}_{k,0}) = 0$, and $\beta(z_{k,j}) = \beta(z_{k,0})$ and $\beta(\overline{z}_{k,j}) = \beta(\overline{z}_{k,0})$ for all $j \in [m]$.

For each $j \in [m]$ let C'_j be a clause obtained from C_j as follows:

- If C_j contains at least one positive literal and at least one negative literal, then C'_j is obtained from C_j by replacing each occurrence of a variable z_k (for $k \in [n]$) with the variable $z_{k,j}$ (for example, $C_j = \{z_1, z_2, \neg z_3\}$ yields $C'_j = \{z_{1,j}, z_{2,j}, \neg z_{3,j}\}$).

- If C_j contains only positive literals, then C'_j is obtained by choosing an arbitrary $k \in [n]$ with $z_k \in C_j$ and letting C'_j be obtained from C_j by replacing z_k with the negative literal $\neg\overline{z}_{k,j}$ and replacing every literal z_ℓ in C_j (with $\ell \in [n]$, $\ell \neq k$) with the positive literal $z_{\ell,j}$.
- If C_j contains only negative literals, then C'_j is obtained by choosing an arbitrary $k \in [n]$ with $\neg z_k \in C_j$ and letting C'_j be obtained from C_j by replacing $\neg z_k$ with the positive literal $\overline{z}_{k,j}$ and replacing every literal $\neg z_\ell$ in C_j (with $\ell \in [n]$, $\ell \neq k$) with the negative literal $\neg z_{\ell,j}$.

Let $\mathscr{C}' := \{C'_1, \ldots, C'_m\}$ and let $f(\mathscr{C}) := \mathscr{C}' \cup \mathscr{K}$. It is straightforward to verify the following:

(1) $f(\mathscr{C})$ satisfies the conditions (1) and (2) of Definition 3.14, i.e., it is a Special-3SAT-instance.
(2) There exists an assignment $\beta\colon Vars(f(\mathscr{C})) \to \{0,1\}$ that satisfies $f(\mathscr{C})$ if, and only if, there exists an assignment $\alpha\colon Vars(\mathscr{C}) \to \{0,1\}$ that satisfies $\mathscr{C}$.
(3) Upon input of $\mathscr{C}$, the set $f(\mathscr{C})$ can be constructed in polynomial time.

In summary, f is a polynomial-time reduction from 3SAT to Special-3SAT, completing the proof of Lemma 3.15. □

Proof of Claim 3.17.
Let $P \in Sol(G,w)$. From Theorem 3.7 we obtain that P is a 1:N / N:1 matching for G with $|X| \geqslant 2$ for all $X \in P$. Let $h := h_P$ and $\ell := \ell_P$. Clearly,

$$w(G_P) \;=\; \tfrac{h+0.5\ell}{h+\ell} \;=\; 0.5 \,+\, 0.5\cdot\tfrac{h}{h+\ell}.$$

Note that $\ell \geqslant 2m$, because of the following reasoning: For each of the nodes $c_{i,j}$ with $i \in [2]$ and $j \in [m]$ the graph G_P must contain an edge e with $c_{i,j} \in e$. Each edge $e \in E(G)$ with $c_{i,j} \in e$ is *light*; and no such edge contains another node in $\{c_{i',j'} : i' \in [2], j' \in [m]\}$. Thus, $E(G_P)$ contains at least $2m$ light edges, i.e., $\ell \geqslant 2m$. Hence,

$$w(G_P) \;=\; 0.5 \,+\, 0.5\cdot\tfrac{h}{h+\ell} \;\leqslant\; 0.5 \,+\, 0.5\cdot\tfrac{h}{h+2m} \;=\; 1 - \tfrac{m}{h+2m}.$$

Note that $h \leqslant n+m$, because $E(G)$ contains only $n+m$ heavy edges. Hence,

$$w(G_P) \;\leqslant\; 1 - \tfrac{m}{h+2m} \;\leqslant\; 1 - \tfrac{m}{n+m+2m} \;=\; \tfrac{n+2m}{n+3m} \;=\; q.$$

This proves the first statement of the claim. The second statement of Claim 3.17 is obtained by observing that in the above equations, all inequalities "$\leqslant$" can be replaced by equalities "=" if, and only if, $\ell = 2m$ and $h = n+m$. This completes the proof of Claim 3.17. □

Proof of Claim 3.18.
Let $P \in Sol(G,w)$ with $w(G_P) = q$. From Claim 3.17 we obtain that $h_P = n+m$ and $\ell_P = 2m$. In particular, $E(G_P)$ contains every heavy edge of (G,w), i.e., it contains the edges $e_{z,k}$ and $e_{d,j}$ for all $k \in [n]$ and $j \in [m]$. From Theorem 3.7 we know that P is a 1:N / N:1 matching with $|X| \geqslant 2$ for all $X \in P$.

Consider an arbitrary $k \in [n]$ and let X_k be the element in P with $v_{1,k} \in X_k$. Since $e_{z,k} = \{v_{1,k}, v_{2,k}\} \in E(G_P)$, we have: $v_{2,k} \in X_k$. From Definition 3.6 we obtain that there exists an $i_k \in [2]$ such that all elements in X_k, except from $v_{i_k,k}$, belong to $V_{3-i_k}(G)$. By definition of G we obtain:

$(*)$: all elements in $X_k \setminus \{v_{1,k}, v_{2,k}\}$ are of the form $c_{3-i_k,j}$ where $j \in [m]$ such that $e_{i_k,k,j} \in E(G)$.

For all $i \in [2]$ and $j \in [m]$ let $X_{i,j}$ be the element in P with $c_{i,j} \in X_{i,j}$.

Case 1: For the considered k, there exists a $j \in [m]$ such that $X_{1,j} = X_k$. In this case, we obtain from $(*)$ that $i_k = 2$ and $e_{2,k,j} = \{v_{2,k}, c_{1,j}\} \in E(G)$, i.e., clause C_j contains the negative literal $\neg z_k$. Furthermore, for all $j' \in [m]$ with $X_{1,j'} = X_k$, clause $C_{j'}$ contains the negative literal $\neg z_k$.

In this case we let $\alpha(z_k) = 0$, and this ensures that all clauses $C_j \in \mathscr{C}$ that contain the negative literal $\neg z_k$ are satisfied.

Case 2: For the considered k, Case 1 does not apply, but there exists a $j \in [m]$ such that $X_{2,j} = X_k$. With the same reasoning as in Case 1 we obtain that C_j contains the positive literal z_k; and, furthermore, for all $j' \in [m]$ with $X_{2,j'} = X_k$, clause $C_{j'}$ contains the positive literal z_k.

In this case we let $\alpha(z_k) = 1$, and this ensures that all clauses $C_j \in \mathscr{C}$ that contain the positive literal z_k are satisfied.

Case 3: None of the Cases 1 or 2 applies to the considered k. Then, for each $j \in [m]$, the set $X_{1,j}$ contains none of the nodes $v_{i,k}$ for $i \in [2]$ and $k \in [n]$. But $|X_{1,j}| \geqslant 2$ and $c_{1,j} \in X_{1,j}$, and hence $X_{1,j}$ must contain a neighbor of $c_{1,j}$ in G. By our definition of G, the only available neighbor is $d_{2,j}$. Hence, $X_{1,j} \supseteq \{c_{1,j}, d_{2,j}\}$. We already know that $e_{d,j} = \{d_{1,j}, d_{2,j}\} \in E(G_P)$; hence $X_{1,j} \supseteq \{c_{1,j}, d_{1,j}, d_{2,j}\}$. With the same reasoning we also obtain that $X_{2,j} \supseteq \{c_{2,j}, d_{1,j}, d_{2,j}\}$. Since the elements in P are pairwise disjoint, we obtain that $X_{1,j} = X_{2,j} \supseteq \{c_{1,j}, c_{2,j}, d_{1,j}, d_{2,j}\}$. Thus, this set contains at least two nodes from each of the sets $V_1(G)$ and $V_2(G)$. Hence P is not a 1:N / N:1 matching, a contradiction to our choice of P. Hence, Case 3 cannot apply.

In summary, for each $k \in [n]$ either Case 1 or Case 2 applies. Let $\alpha\colon \mathit{Vars}(\mathscr{C}) \to \{0,1\}$ be the assignment defined according to the Cases 1 and 2. To complete the proof of Claim 3.18 it suffices to show that α satisfies $\mathscr{C}$. Consider an arbitrary $j' \in [m]$.

If $X_{1,j'} = X_k$ for some $k \in [n]$ then, according to Case 1, clause $C_{j'}$ contains the negative literal $\neg z_k$ and $\alpha(z_k) = 0$, and hence α satisfies clause $C_{j'}$.

Otherwise, if $X_{2,j'} = X_k$ for some $k \in [n]$ then, according to Case 2, clause $C_{j'}$ contains the positive literal z_k and $\alpha(z_k) = 1$, and hence α satisfies clause $C_{j'}$.

The only remaining case is that $X_{1,j'}$ and $X_{2,j'}$ contain none of the nodes $v_{i,k}$ for $i \in [2]$ and $k \in [n]$. In Case 3 above, we have already shown that this leads to a contradiction. In summary, we obtain that α satisfies $\mathscr{C}$. This completes the proof of Claim 3.18. □

Proof of Claim 3.19.

Let $\alpha\colon \mathit{Vars}(\mathscr{C}) \to \{0,1\}$ be an assignment that satisfies $\mathscr{C}$. For every $j \in [m]$, let us fix an arbitrary $k_j \in [n]$ such that the following holds: $\big(\ \alpha(z_{k_j}) = 0$ and $\neg z_{k_j} \in C_j\ \big)$ or $\big(\ \alpha(z_{k_j}) = 1$ and $z_{k_j} \in C_j\ \big)$; such a k_j exists because α satisfies the clause C_j.

Now, let G' be the subgraph of G with $V_i(G') := V_i(G)$ for $i \in [2]$, and where $E(G')$ consists of the following edges:

- all *heavy* edges of G,
- for every $j \in [m]$ with $\alpha(z_{k_j}) = 1$, the edges $e_{1,k_j,j} = \{v_{1,k_j}, c_{2,j}\}$ and $e_{1,cd,j} = \{c_{1,j}, d_{2,j}\}$ (note that $e_{1,k_j,j} \in E(G)$ by our choice of k_j),
- for every $j \in [m]$ with $\alpha(z_{k_j}) = 0$, the edges $e_{2,k_j,j} = \{v_{2,k_j}, c_{1,j}\}$ and $e_{2,cd,j} = \{c_{2,j}, d_{1,j}\}$ (note that $e_{2,k_j,j} \in E(G)$ by our choice of k_j).

Note that $E(G')$ consists of $n{+}m$ heavy edges and $2m$ light edges; thus $w(G') = \frac{n+2m}{n+3m} = q$.

Let P be the partition of $V(G') = V_1(G) \cup V_2(G)$ which, for each connected component of G', contains an element X consisting of the nodes that belong to this connected component. It is not difficult to verify that $|X| \geqslant 2$ for all $X \in P$, that $G' = G_P$, and that P is a 1:N / N:1 matching for G. From Theorem 3.7 we obtain that $P \in \mathit{Sol}(G, w)$. We have already seen that $w(G_P) = w(G') = q$. This completes the proof of Claim 3.19. □

Received December 2024; revised February 2025; accepted March 2025