Minimizing AOD Trap Activations for Atom Shuttling in Neutral-Atom Quantum Architectures

Francesco Decataldo[1,*], Christian Bianchini[1,*] and Enrico Santi[1,*]

[1]University of Udine, Via delle Scienze 206, Udine, 33100, Italy

Abstract

Neutral-atom quantum computers store quantum information in atoms confined by optical traps. Unlike superconducting architectures, neutral-atom platforms can shuttle atoms across the processor using acousto-optic deflector (AOD) movable traps to shuttle atoms before gate execution. When compiling quantum circuits for neutral-atom devices, minimizing the number of AOD traps activations is crucial, since each activation increases the physical overhead of the computation.

We introduce a matrix-based optimization problem that models the AOD traps activation scheme and prove that finding an optimal solution is an NP-hard problem. As a consequence, determining the minimum number of AOD activations required to shuttle a given set of atoms is also NP-hard. To solve practical instances of the problem, we encode it as an Answer Set Programming problem and evaluate the proposed approach experimentally on a set of benchmark instances. We further establish a correspondence between our formulation and Boolean Matrix Factorization (BMF), which opens the way to reusing existing BMF techniques for computing AOD activation schemes.

Keywords

Complexity Theory, Quantum Computing, Quantum Compilation, Optimization

1. Introduction

Neutral-atom quantum computers are a promising candidate for scalable quantum computation [1, 2]. In these devices, qubits are encoded in individual atoms trapped by optical tweezers and arranged on a two-dimensional array of sites (e.g. a $n \times m$ matrix). A distinctive feature of this architecture is that atoms are not necessarily fixed throughout the computation: they can be physically rearranged in order to bring selected qubits into interaction zones, enabling the implementation of multi-qubit gates between atoms that are far apart in the initial layout. This reconfigurability introduces a new compilation problem: the compiler must decide not only where qubits are initially placed, but also which AOD activations move the required atoms with minimal physical overhead [3].

Neutral-atom architectures use two kinds of optical traps [4]. Static traps, generated for example by spatial light modulators (SLMs) define the fixed storage locations in the array. Atoms held in these traps remain to their assigned grid cells unless they are transferred into movable traps. Movable traps are commonly generated using acousto-optic deflectors (AODs). An AOD does not select arbitrary individual grid cells independently. Instead, an AOD activation selecting rows $\mathcal{R}$ and columns $\mathcal{C}$ creates movable traps at all intersections of those rows and columns [3]. Therefore, a single AOD activation involves all the atoms whose location is contained in the Cartesian product $\mathcal{R} \times \mathcal{C}$. Any atoms located at cells in this Cartesian product are affected by the activation. This Cartesian-product structure imposes an important constraint on atom shuttling: at a given compilation step, some atoms may need to be picked up and transported, while other atoms must remain fixed because they are not involved in the current operation. Activating rows $\mathcal{R}$ and columns $\mathcal{C}$ may unintentionally trap or move atoms that should remain fixed, as shown in Figure 1. Consequently, the compiler must choose AOD activations that cover the atoms to be moved while avoiding all atoms that must remain stationary.

ICTCS 2026: Italian Conference on Theoretical Computer Science, September 7–9, 2026, Udine, Italy

*These authors contributed equally.

decataldo.francesco@spes.uniud.it (F. Decataldo); bianchini.christian@spes.uniud.it (C. Bianchini); enrico.santi@uniud.it (E. Santi)

0009-0006-7349-391X (F. Decataldo); 0009-0006-2848-2517 (C. Bianchini); 0009-0008-3556-7177 (E. Santi)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

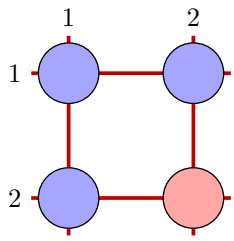

Figure 1: The blue atoms must be shuttled, while the red atom must remain fixed. The AOD activation shown in figure (red lines) is given by $\mathcal{R} = \{1, 2\}$ and $\mathcal{C} = \{1, 2\}$. The activation is not valid since a red atom is affected by the activation ($(2, 2) \in \mathcal{R} \times \mathcal{C}$).

In typical neutral-atom compilation workflows [5, 6], each rearrangement step identifies a set of atoms that must be shuttled so that the required qubit interactions can be performed. At this stage, grouping the necessary pickups into as few valid AOD activations as possible is a natural optimization objective, since each additional activation introduces an overhead. The problem considered in this work arises precisely at this compilation stage: we search for the minimum number of valid AOD activations required to pick up all the atoms that must be moved without disturbing the atoms that are not involved in the current compilation step.

In this paper, we isolate and study this problem as a matrix-cover optimization problem. We represent the atom array by a matrix with entries in $\{1, 0, *\}$, where a 1 denotes an atom that must be moved, a 0 denotes an atom that must remain fixed, and $*$ denotes an empty site. An AOD activation is modeled as the Cartesian product of a set of rows and a set of columns. The goal is to cover all 1 entries using the minimum number of activations, subject to the constraint that no activation covers any 0 entry. We call the resulting optimization problem AOD-Cover, and we also consider its decision variant, k-AOD-Cover.

Quantum compilation for neutral-atom architectures has recently received increasing attention [7, 2]. Existing compilers for reconfigurable atom arrays address problems such as qubit mapping, atom movement, and gate scheduling, exploiting the ability to rearrange atoms during computation in order to reduce routing overhead. Examples include a compiler for zoned neutral-atom architectures [6]. Our work is complementary to existing compilers: rather than proposing a full compilation pipeline, we isolate a lower-level optimization primitive arising during atom shuttling: the problem of minimizing the number of valid AOD activations needed to move a prescribed set of atoms while avoiding atoms that must remain fixed.

Our contribution is threefold. First, we provide a formal abstraction of AOD trap activations as a constrained matrix-cover problem. Second, we prove that k-AOD-Cover is NP-complete by a direct reduction from the Boolean matrix factorization problem [8, 9]; consequently, AOD-Cover is NP-hard. Third, we present an Answer Set Programming (ASP) [10] encoding for computing optimal activation schemes and report preliminary experimental results on benchmark instances.

The rest of the paper is organized as follows. Section 2 introduces the formal definition of AOD-Cover and its decision version. Section 3 proves the computational hardness of the problem. Section 4 presents the ASP encoding and experimental results. Section 5 presents the conclusions outlining also some directions for future work.

2. Problem formalization

We now formalize the optimization problem induced by the AOD activation model described in Section 1. Let $M \in \{1, 0, *\}^{n \times m}$ be a matrix representing an atom array with n rows and m columns. For each

position (r, c), with $r \in \{1, \ldots, n\}$ and $c \in \{1, \ldots, m\}$, the value of $M[r, c]$ has the following meaning:

$$M[r,c] = \begin{cases} 1 & \text{if the atom at position } (r,c) \text{ must be moved.} \\ 0 & \text{if the atom at position } (r,c) \text{ must remain fixed.} \\ * & \text{if the position is empty.} \end{cases}$$

An AOD activation is modeled by a pair $(\mathcal{R}, \mathcal{C})$, where

$$\mathcal{R} \subseteq \{1, \ldots, n\} \qquad \text{and} \qquad \mathcal{C} \subseteq \{1, \ldots, m\}$$

are, respectively, the selected rows and columns. The positions selected by the activation are exactly the cells in the Cartesian product $\mathcal{R} \times \mathcal{C}$. Thus, the activation affects a cell (r, c) if and only if $r \in \mathcal{R}$ and $c \in \mathcal{C}$. We are interested in activations that do not contain any position with an atom that should not be moved.

Definition 1 (Valid activation)**.** *Given a matrix $M \in \{1, 0, *\}^{n \times m}$, an activation $(\mathcal{R}, \mathcal{C})$ is* valid *for M if and only if it does not select any cell that must remain fixed, namely:*

$$M[r, c] \neq 0 \quad \textit{for every } (r, c) \in \mathcal{R} \times \mathcal{C}.$$

The optimization problem consists in finding the minimum number of activations needed to "cover" a given matrix. The formal description is given by Definition 2.

Definition 2 (Valid cover)**.** *A set of activations*

$$\mathcal{S} = \{(\mathcal{R}_1, \mathcal{C}_1), \ldots, (\mathcal{R}_k, \mathcal{C}_k)\}$$

is a valid cover *for M if the following two conditions hold:*

1. *every cell containing an atom that must be moved is covered by at least one activation, that is, for every (r, c) such that $M[r, c] = 1$, there exists $i \in \{1, \ldots, k\}$ such that*

$$(r, c) \in \mathcal{R}_i \times \mathcal{C}_i.$$

2. *every activation in $\mathcal{S}$ is valid for M, that is, for every $i \in \{1, \ldots, k\}$ we have*

$$M[r, c] \neq 0 \quad \textit{for every } (r, c) \in \mathcal{R}_i \times \mathcal{C}_i.$$

The formal definition of the optimization problem is, then, the following.

Definition 3 (AOD-Cover)**.** *Given a matrix $M \in \{1, 0, *\}^{n \times m}$, find the smallest set $\mathcal{S}$ of activations such that it is a valid cover for the matrix M.*

The corresponding decision problem is defined as follows.

Definition 4 (k-AOD-Cover)**.** *Given a matrix $M \in \{1, 0, *\}^{n \times m}$ and $k \in \mathbb{N}$, deciding if there exists a set $\mathcal{S}$ of at most k activations that is a valid cover for the matrix M.*

Observation 1. Since one feasible (non optimal) solution is to cover each 1 with its own 1×1 submatrix, it follows that the maximum number of activation needed is at most $n \cdot m$, so it grows polynomially in the size of the input.

Thanks to this observation, the decision variant of the problem can be used, in conjunction with a binary search, to compute the minimum value for k; hence, studying the decision version is sufficient to characterize the complexity of the optimization problem.

3. NP-completeness

We prove that k-AOD-Cover, the decision version of AOD-Cover is NP-complete. The proof is by a polynomial-time reduction from the decision version of Boolean Matrix Factorization (BMF). An instance of BMF consists of a $n \times m$ Boolean matrix $M \in \{0,1\}^{n \times m}$ and an integer k. The question is whether there exist two Boolean matrices $V \in \{0,1\}^{n \times k}$ and $H \in \{0,1\}^{k \times m}$ such that:

$$V \cdot H = M$$

where multiplication and addition between entries are, respectively, conjunction and disjunction. We say that $\langle V, H \rangle$ is a Boolean factorization of M. This problem was proven to be NP-Complete in [11].

Before showing the reduction, we introduce a key observation about a Boolean factorization. The matrix V can be seen as a set of k binary column vectors of size $n \times 1$. At the same time, the matrix H is simply a collection of k binary row vectors of size $1 \times m$. Describing the two matrices as $V = (V_1, \ldots, V_k)$ and $H = \left(H^1, \ldots, H^k\right)$, we can rewrite their product as

$$V \cdot H = \sum_{i=1}^{k} V_i \cdot H^i$$

where each term $V_i \cdot H^i$ is a Boolean outer product of the two vectors and the sum is the boolean sum (i.e. the logical inclusive or). This description of the matrix factorization yields a useful result. Indeed, multiplying the vectors V_i and H^i amounts to setting to 1 the elements at the intersection between the rows of the 1-elements of V_i and the columns of the 1-elements of H^i. At the same time, since the sum of the outer products is performed using the boolean sum, the resulting matrix has a certain entry set to 1 if and only if one of the outer product matrices has a 1 in the same entry. This means that:

Observation 2. Given a binary factorization $\langle V, H \rangle$ of M, then, for every $1 \leq r \leq n$ and $1 \leq c \leq m$,

$$M[r,c] = 1 \iff \exists i \left(V_i[r] = 1 \wedge H^i[c] = 1\right).$$

Thus each outer product selects the Cartesian product of the rows where V_i is 1 and the columns where H^i is 1, exactly matching the structure of one AOD activation.

3.1. Reduction from BMF to k-AOD-Cover

We will show that BMF instances are equivalent to AOD-Cover instances that do not contain any $*$. Since both problems take a $n \times m$ binary matrix as input, it is sufficient to show that any valid cover of the matrix M corresponds to a binary matrix factorization, and vice versa.

We start by defining the factorization encoded by an AOD cover. Given a cover $\mathcal{S}$ of size k, we define the factorization $f(\mathcal{S}) = \langle V, H \rangle$ as the matrices $V = (V_1, \ldots, V_k)$ and $H = (H^1, \ldots, H^k)$ such that $V_i[r] = 1$ if and only if $r \in \mathcal{R}_i$ and $H^i[c] = 1$ if and only if $c \in \mathcal{C}_i$.

We also describe, in the same way, the cover encoded by a factorization. Given a factorization $\langle V, H \rangle$ of size k of M, we define the cover $g(\langle V, H \rangle) = \{(\mathcal{R}_1, \mathcal{C}_1), \ldots, (\mathcal{R}_k, \mathcal{C}_k)\}$ such that $r \in \mathcal{R}_i$ if and only if $V_i[r] = 1$ and $c \in \mathcal{C}_i$ if and only if $H^i[c] = 1$.

Lemma 1. *Given a binary $n \times m$ matrix M and a valid cover $\mathcal{S} = \{(\mathcal{R}_1, \mathcal{C}_1), \ldots, (\mathcal{R}_k, \mathcal{C}_k)\}$, the factorization $f(\mathcal{S}) = \langle V, H \rangle$ is a valid factorization of M (i.e. $V \cdot H = M$).*

Proof. The proof is straightforward, since:

$$M[r,c] = 1 \iff (r,c) \in \mathcal{R}_i \times \mathcal{C}_i \iff r \in \mathcal{R}_i \wedge c \in \mathcal{C}_i \iff V_i[r] = 1 \wedge H^i[c] = 1$$

for some $1 \leq i \leq k$, which, thanks to Observation 2, proves that $\langle V, H \rangle$ is a binary factorization of M. □

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \Big\}\mathcal{R}_1 \; \begin{pmatrix} \underbrace{1 \quad 1}_{\mathcal{C}_1} & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \Big\}\mathcal{R}_2 \; \begin{pmatrix} 0 & 0 & \underbrace{1}_{\mathcal{C}_2} \end{pmatrix}$$

Figure 2: Illustration of the correspondence between AOD-Cover and Boolean Matrix Factorization problem. The red and green patterns represent both the factors, and a valid cover for the original matrix, given by $\{(\mathcal{R}_1, \mathcal{C}_1), (\mathcal{R}_2, \mathcal{C}_2)\}$

With the same argument, one can also prove that the cover encoded by a valid binary factorization is also valid.

Lemma 2. *Given a binary $n \times m$ matrix M and a valid factorization $\langle V, H \rangle$ of size k, the cover $g(\langle V, H \rangle) = \{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$ is valid for M.*

We now prove the NP-completeness of k-AOD-Cover.

Theorem 3. *k-AOD-Cover is NP-complete.*

Proof. To prove the NP-completeness of k-AOD-Cover, we first prove that the problem belongs to NP and then show a polynomial-time reduction from BMF.

NP First, k-AOD-Cover belongs to NP. A certificate consists of $l \leq k$ activations

$$\{(\mathcal{R}_1, \mathcal{C}_1), \ldots, (\mathcal{R}_l, \mathcal{C}_l)\}.$$

Notice that, thanks to Observation 1, the size of a certificate is at most $(n + m) \cdot nm$, so it is guaranteed to be polynomial in the size of the input. In polynomial time, we can check that every activation avoids all entries equal to 0, and that every entry equal to 1 is covered by at least one activation.

NP-Hardness From Lemmas 1 and 2 we obtain that an instance of BMF has a factorization of rank $l \leq k$ if and only if the same matrix admits a valid AOD-Cover of size $l \leq k$. From this result and from the NP-hardness of BMF, we prove that k-AOD-Cover is NP-hard.

Since it is both NP-hard and also in NP, it is NP-complete. □

An illustration of the reduction is shown in Figure 2.

Corollary 1. *The corresponding optimization problem AOD-Cover (Definition 3) is NP-hard.*

3.2. Reduction from k-AOD-Cover to BMF

In the previous section, we proved that k-AOD-Cover is NP-complete by giving a polynomial-time reduction from BMF to k-AOD-Cover. We now show a reduction in the opposite direction, from k-AOD-Cover to BMF. Although this reduction is not needed for the NP-completeness proof, it is useful from a practical point of view. Boolean Matrix Factorization is a well-studied problem [9], and several algorithmic approaches and tools for solving BMF instances have been developed in recent years [12, 13]. Therefore, by reducing k-AOD-Cover to BMF, one can reuse existing BMF solvers to compute valid AOD covers.

To do so we encode a k-AOD-Cover instance into BMF in the following way. Given a matrix $M \in \{1, 0, *\}^{n \times m}$, let

$$P = \{p = (r, c) \mid M[r, c] = *\}$$

be the set of positions containing a $*$-entry. We construct a Boolean matrix

$$M' \in \{0,1\}^{(n+|P|)\times(m+|P|)}.$$

The first n rows of M' are called the *original rows*, and are indexed by $r \in \{1,\dots,n\}$ The first m columns of M' are called the *original columns*, and are indexed by $c \in \{1,\dots,m\}$. For every $p \in P$, we add one new row α_p and one new column β_p. On the original rows and columns we set

$$M'[r,c] = \begin{cases} 1 & \text{if } M[r,c] = 1 \vee M[r,c] = * \\ 0 & \text{if } M[r,c] = 0 \end{cases}$$

Moreover, for every $p = (r_p, c_p) \in P$, we additionally set

$$M'[r_p, \beta_p] = 1 \qquad M'[\alpha_p, c_p] = 1 \qquad M'[\alpha_p, \beta_p] = 1$$

All remaining non original entries are set to 0. As an example, consider the following encoding:

$$M = \begin{pmatrix} 1 & *_1 & 0 \\ 0 & 1 & *_2 \\ *_3 & 1 & 0 \end{pmatrix} \qquad \Longrightarrow \qquad M' = \begin{array}{c} \\ \\ \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{array} \begin{array}{c} \begin{array}{ccc|ccc} \beta_1 & \beta_2 & \beta_3 \end{array} \\ \left(\begin{array}{ccc|ccc} 1 & 1_1 & 0 & 1_1 & 0 & 0 \\ 0 & 1 & 1_2 & 0 & 1_2 & 0 \\ 1_3 & 1 & 0 & 0 & 0 & 1_3 \\ \hline 0 & 1_1 & 0 & 1_1 & 0 & 0 \\ 0 & 0 & 1_2 & 0 & 1_2 & 0 \\ 1_3 & 0 & 0 & 0 & 0 & 1_3 \end{array}\right) \end{array}$$

Thus, each $*$-entry of M gives rise to a 2×2 all-1s block inside M'

	c_p	β_p
r_p	1	1
α_p	1	1

The row α_p and column β_p have no other 1-entries.

We now prove that this construction is correct: solving the resulting BMF instance $(M', k + |P|)$ allows us to decide the original k-AOD-Cover instance (M, k).

Theorem 4. *(M, k) is a yes-instance of k-AOD-Cover if and only if $(M', k + |P|)$ is a yes instance of BMF.*

Proof. We need to prove two directions:

($\Rightarrow$) Assume that (M, k) is a yes-instance of k-AOD-Cover. Then there exists a valid cover

$$\mathcal{S} = \{(\mathcal{R}_1, \mathcal{C}_1), \dots, (\mathcal{R}_l, \mathcal{C}_l)\}$$

of size $l \le k$. For each activation $(\mathcal{R}_i, \mathcal{C}_i)$, consider the same rectangle $\mathcal{R}_i \times \mathcal{C}_i$ in the original rows and columns of M'. Since the activation is valid for M, it does not contain any entry (r, c) such that $M[r, c] = 0$. Therefore, the corresponding rectangle contains only 1-entries of M'. These l rectangles cover all entries of M' corresponding to 1-entries of M. It remains to cover the entries introduced for the $*$-positions. For every $p = (r_p, c_p) \in P$, add the rectangle

$$\{r_p, \alpha_p\} \times \{c_p, \beta_p\}.$$

By construction, this is an all-one rectangle of M', since all four entries

$$(r_p, c_p), \quad (r_p, \beta_p), \quad (\alpha_p, c_p), \quad (\alpha_p, \beta_p)$$

are equal to 1 in M'. Therefore, all 1-entries of M' are covered by at most

$$l + |P| \le k + |P|$$

rectangles. Equivalently, M' admits a Boolean matrix factorization of rank at most $k + |P|$. Hence $(M', k + |P|)$ is a yes-instance of BMF.

(⇐) Assume that $(M', k + |P|)$ is a yes-instance of BMF. Then M' has a Boolean factorization of rank at most $k + |P|$. Equivalently, by the equivalence between matrix factorization and covering rectangles, the 1-entries of M' can be covered by at most $k + |P|$ rectangles. Consider any $p = (r_p, c_p) \in P$. The entry (α_p, β_p) is equal to 1, so it must be covered by some rectangle. We claim that any rectangle covering (α_p, β_p) is contained in

$$\{r_p, \alpha_p\} \times \{c_p, \beta_p\}.$$

Indeed, the column β_p contains 1 only in rows r_p and α_p. Similarly, the row α_p contains 1 only in columns c_p and β_p. Thus the claim follows. Moreover, no single rectangle can cover two entries (α_p, β_p) and (α_q, β_q) with $p \neq q$. Indeed, such a rectangle would also contain the cross-entry (α_p, β_q), which is equal to 0 by construction.

Therefore, the $|P|$ entries require at least $|P|$ distinct rectangles. Choose one rectangle covering each entry and remove these $|P|$ rectangles. Since each removed rectangle is contained in the corresponding 2×2 block

$$\{r_p, \alpha_p\} \times \{c_p, \beta_p\},$$

the only original entry it can cover is (r_p, c_p), which corresponds to a $*$-entry of M. Hence the removed rectangles do not cover any original entry (r, c) with $M[r, c] = 1$. At this point, at most $(k + |P|) - |P| = k$ rectangles remains. Restrict each remaining rectangle to the original rows and columns of M. These rectangles must cover all the 1s M. Therefore, these last (at most k) restricted rectangles form a valid cover for M and (M, k) is a yes-instance of k-AOD-Cover.

We have shown that

$$(M, k) \in k\text{-AOD-Cover} \iff (M', k + |P|) \in \text{BMF}.$$

Therefore, k-AOD-Cover reduces to BMF in polynomial time.

□

3.3. How to solve the optimization problem

Given that the problem has been proven to be NP-complete, we experimented solving it by encoding instances in *Answer set programming*, a flexible declarative formalism widely used to model combinatorial search and optimization problems [14]. For simplicity, we encode a matrix by specifying the value of a cell (r, c) to 1 if the atom must be moved, 0 if it must remain fixed, or -1 if the position is empty. We now present the ASP program used to solve the problem:

```
row(0..M-1) :- dim(M,N).
col(0..N-1) :- dim(M,N).

% One activation per 1-cell suffices in the worst case (define an upper bound)
block_count(K) :- K = #count { R,C : cell(R,C,1) }.
block(1..K)    :- block_count(K).

% Each activation independently chooses a non-empty row-set and col-set
1 { row_in(B,R) : row(R) } :- block(B).
1 { col_in(B,C) : col(C) } :- block(B).

% An activation is invalid iff its Cartesian product contains a 0-cell
invalid(B) :- block(B), row_in(B,R), col_in(B,C), cell(R,C,0).
valid(B)   :- block(B), not invalid(B).

0 { used(B) } 1 :- block(B), valid(B).

% A used activation covers the 1-cells in its Cartesian product
covers(B,R,C) :- used(B), row_in(B,R), col_in(B,C), cell(R,C,1).

% Every 1-cell must be covered to be a solution
:- cell(R,C,1), not covers(_,R,C).

```

```
% --- Symmetry breaking ---

% Use lowest block ids first. If B+1 is used then B must be used, eliminates equivalent labellings
:- block(B), used(B+1), not used(B).

% No two used blocks may have identical row-set AND col-set. i.e. don't consider twice a same
    submatrix
% A duplicate activation is always redundant
:- used(B1), used(B2), B2 > B1,
   #count { R : row_in(B1,R), not row_in(B2,R) } = 0,
   #count { R : row_in(B2,R), not row_in(B1,R) } = 0,
   #count { C : col_in(B1,C), not col_in(B2,C) } = 0,
   #count { C : col_in(B2,C), not col_in(B1,C) } = 0.

% Minimize number of used activations
#minimize { 1,B : used(B) }.

% Show the number and the submatrices
#show used/1.
#show row_in/2.
#show col_in/2.
```

A problem instance can be easily encoded by declaring a fact `dim` specifying the number of rows and columns of the matrix, and defining the matrix itself via a ternary predicate `cell`. The arguments of `cell` are respectively the row, column of the cell and the value contained.
The following example encodes a simple 2×3 matrix:

```
%  1  0  1
%  1 -1  0

dim(2,3).
cell(0,0, 1). cell(0,1, 0). cell(0,2, 1).
cell(1,0, 1). cell(1,1,-1). cell(1,2, 0).
```

4. Experimental results

We now report the results obtained on different AOD-Cover instances by using the model presented in Section 3.3. All the tests[1] have been executed on a system equipped with an Intel Core i7-13700KF, 128 GB of DDR4 RAM and Clingo version 5.6.2.

A total of 180 instances have been randomly generated, subdivided into 9 distinct batches of tests presenting the same size. The instances range from matrices of size 5×5 to 18×17. Batches of 20 instances per size have been used due to the high variability of the solving times. A timeout of 20 minutes has been set for each test instance. Figure 3 presents the average times and standard deviation for the different test batches. Blue dots in the plot represent the average runtime of the corresponding batch of instances. Red triangles represent the average runtime of the non timed-out instances in the batch, while also signaling that timeouts occurred in that batch. Note that batches where all instances resulted in a timeout (e.g. instances of size 306, composed by matrices 18×17) are omitted from the plot. Figure 4 presents the average amount and standard deviation of memory used for the different tests.
While presenting a noticeable variance in the runtime (see Figure 3), without the symmetry breaking constraints included in the program reported in Section 3.3 all the test instances of size 160 resulted in a timeout [15]. On the other hand, as shown in Figure 4, even when considering timed-out instances the standard deviation of memory usage remains relatively low.

5. Conclusions

In this paper we introduced AOD-Cover, a matrix-based optimization problem that captures a fundamental minimization task arising in atom shuttling for neutral-atom quantum architectures. In our

[1]The scripts and code used to perform the test are available on GitHub (https://github.com/fradeca01/Min-AOD-cover).

Table 1
Summary statistics by instance batch. The table includes the number of solved and timed-out instances for each batch as well as memory and runtime average values. Memory statistics are computed over all instances (including timeouts), while runtime statistics are computed only over solved instances in each batch. Since no instance of size 306 has been solved, no average runtime is reported.

Instance matrix	Size	Instances	Solved	Timeouts	Avg. Mem. (MB)	Avg. Time (s)
5x5	25	20	20	0	9.84	0.01
5x7	35	20	20	0	11.7	0.02
8x8	64	20	20	0	24.4	0.1
10x10	100	20	20	0	52.7	0.39
10x13	130	20	20	0	98.4	6.2
10x16	160	20	19	1	148	16.4
15x15	225	20	9	11	322	128
15x17	255	20	3	17	447	433
17x18	306	20	0	20	670	–

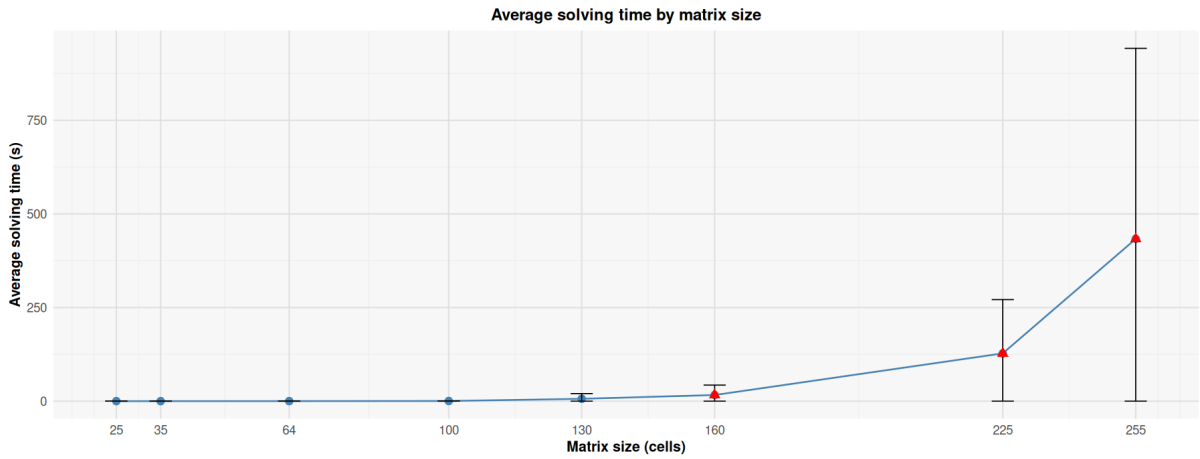

Figure 3: The average solving times according to the instance (matrix) size, bars indicate standard deviation.

model, atoms that must be moved are represented by entries equal to 1, atoms that must remain fixed by entries equal to 0, and empty positions by $*$ entries. AOD activations are modeled as Cartesian products of selected rows and columns and the goal is to cover all atoms to be moved while avoiding all atoms that must remain stationary.

We proved that the decision version of the problem, k-AOD-Cover, is NP-complete by providing a polynomial-time reduction from Boolean Matrix Factorization. This result shows that minimizing the number of AOD trap activations is computationally hard even in our matrix model, and therefore identifies an intrinsic source of difficulty in this part of neutral-atom compilers.

To solve practical instances, we proposed an Answer Set Programming (ASP) encoding that searches for optimal activation schemes. Our experiments show that the encoding can solve small and medium-sized instances exactly, while the runtime grows significantly as the matrix size increases. The results also indicate that symmetry-breaking constraints are important for making the approach effective in practice.

5.1. Future work

Several directions remain open for future developments. First, alternative optimization formulations – such as encodings into MaxSAT instances, using integer programming or using approximation algorithms – may yield better results and faster runtimes. Second, the reduction to Boolean Matrix Factorization suggests that specialized (and well tested) BMF algorithms and solvers could be adapted

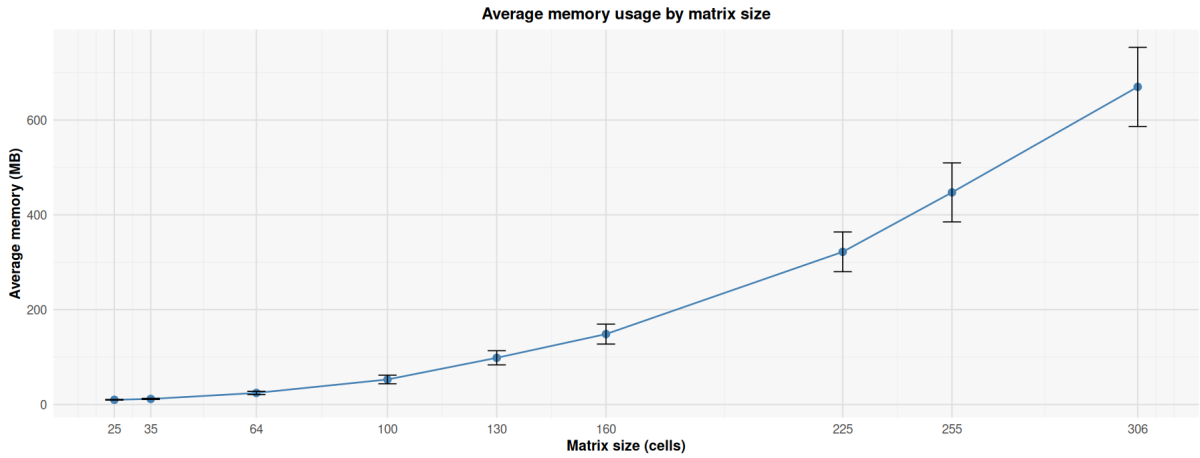

Figure 4: Average memory consumption according to the instance (matrix) size, bars indicate standard deviation.

to solve AOD-Cover instances more efficiently. Third, it would be useful to evaluate the model on instances extracted from real neutral-atom compilation pipelines, rather than only on synthetic benchmarks, to better extrapolate real-world time and memory usages. Finally, the abstraction could be extended to include additional hardware constraints, such as multi-zone scheduling.

Overall, our results provide a formal complexity-theoretic foundation for AOD activation minimization and suggest that this problem could be treated as a dedicated optimization primitive inside future neutral-atom quantum compilers.

Acknowledgments

Francesco Decataldo has been partially supported by INdAM-GNCS project *Algebra lineare quantistica, state preparation e compilazione di circuiti quantistici* (CUP E53C25002010001) and by the regional project QUASAR-FVG *Calcolo e simulazione quantistica: sviluppo, applicaizoni e ricerca in Friuli Venezia Giulia* (CUP G23C25001510002). Enrico Santi is supported by FSE/FVG PhD Grant on "XAI-FVG Explainability of Weather Forecasting in FVG" (G23C23001130008).

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] H. J. Manetsch, G. Nomura, E. Bataille, X. Lv, K. H. Leung, M. Endres, A tweezer array with 6,100 highly coherent atomic qubits, Nature 647 (2025) 60–67.

[2] D. Bluvstein, A. A. Geim, S. H. Li, S. J. Evered, J. P. Bonilla Ataides, G. Baranes, A. Gu, T. Manovitz, M. Xu, M. Kalinowski, et al., A fault-tolerant neutral-atom architecture for universal quantum computation, Nature 649 (2026) 39–46.

[3] L. Schmid, D. F. Locher, M. Rispler, S. Blatt, J. Zeiher, M. Müller, R. Wille, Computational capabilities and compiler development for neutral atom quantum processors—connecting tool developers and hardware experts, Quantum Science and Technology 9 (2024) 033001.

[4] H. Kim, W. Lee, H.-g. Lee, H. Jo, Y. Song, J. Ahn, In situ single-atom array synthesis using dynamic holographic optical tweezers, Nature communications 7 (2016) 13317.

[5] Y. Stade, L. Schmid, L. Burgholzer, R. Wille, An abstract model and efficient routing for logical entangling gates on zoned neutral atom architectures, in: 2024 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 1, IEEE, 2024, pp. 784–795.
[6] Y. Stade, L. Burgholzer, R. Wille, Search smarter, not harder: A scalable, high-quality zoned neutral atom compiler, arXiv preprint arXiv:2512.13790 (2025).
[7] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, et al., Logical quantum processor based on reconfigurable atom arrays, Nature 626 (2024) 58–65.
[8] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, H. Mannila, The discrete basis problem, IEEE transactions on knowledge and data engineering 20 (2008) 1348–1362.
[9] P. Miettinen, S. Neumann, Recent developments in boolean matrix factorization, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI'20, 2021, p. 7.
[10] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Answer set solving in practice, Springer Nature, 2022.
[11] L. J. Stockmeyer, The set basis problem is NP-complete, IBM Thomas J. Watson Research Division, 1975.
[12] O. Günlük, R. A. Hauser, R. Á. Kovács, Binary matrix factorization and completion via integer programming, Mathematics of Operations Research 49 (2024) 1278–1302.
[13] C. Wan, W. Chang, T. Zhao, M. Li, S. Cao, C. Zhang, Fast and efficient boolean matrix factorization by geometric segmentation, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 6086–6093.
[14] A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming: A. falkner et al., KI-Künstliche Intelligenz 32 (2018) 165–176.
[15] C. Drescher, O. Tifrea, T. Walsh, Symmetry-breaking answer set solving, AI Commun. 24 (2011) 177–194. URL: https://doi.org/10.3233/AIC-2011-0495. doi:10.3233/AIC-2011-0495.