Optimizing Dancing Links for the N-Queens Problem Using Primary and Secondary Constraints

Paras Bansal

Department of Computer Science and Engineering

Chandigarh University

Punjab, India

Abstract

The N-Queens problem is a classical combinatorial constraint satisfaction problem that has been extensively studied using backtracking, bitmasking, and exact cover formulations. Dancing Links (DLX), proposed by Donald E. Knuth, is a well-known technique for efficiently solving exact cover problems by enabling constant-time removal and restoration of constraints during backtracking. However, conventional DLX-based formulations for the N-Queens problem treat all constraints—rows, columns, and diagonals—as primary constraints, enforcing each to be satisfied exactly once. This modeling introduces unnecessary branching, as diagonal constraints inherently require only conflict avoidance rather than exact coverage.

In this paper, we propose a constraint-aware modification to the DLX formulation for the N-Queens problem by classifying row and column constraints as primary, while treating diagonal and anti-diagonal constraints as secondary. This distinction more accurately reflects the logical requirements of the problem, ensuring that primary constraints are satisfied exactly once while secondary constraints are used solely for conflict detection. The proposed approach reduces redundant branching and prunes infeasible search paths earlier in the recursion.

Experimental evaluations conducted on board sizes ranging from N=8 to N=16 demonstrate that the proposed method significantly reduces the number of explored nodes and execution time compared to the standard DLX formulation, while preserving correctness. The results confirm that appropriate constraint classification within the DLX framework leads to improved performance and scalability for the N-Queens problem. This work highlights the importance of problem-specific constraint modeling when applying generic exact cover algorithms.

Introduction

The N-Queens problem is a classical combinatorial search problem that requires placing N queens on an N×N chessboard such that no two queens attack each other. This implies that no two queens may share the same row, column, or diagonal. Due to its simple formulation and rapidly growing search space with increasing N, the problem has been widely used as a benchmark for evaluating constraint satisfaction and backtracking algorithms.

Traditional approaches to solving the N-Queens problem include naive backtracking, branch-and-bound techniques, and bitmask-based optimizations. While these methods perform well for moderate board sizes, their scalability becomes limited as N increases. To address these challenges, the problem has also been formulated as an Exact Cover problem, enabling the use of more advanced solving techniques such as Algorithm X and Dancing Links (DLX).

Dancing Links, introduced by Donald E. Knuth, is an efficient technique for implementing Algorithm X using a circular doubly linked list representation. DLX allows constraints to be removed and restored in constant time, making it particularly effective for problems involving heavy backtracking and frequent state changes. As a result, DLX has been successfully applied to problems such as Sudoku, exact cover tiling, and the N-Queens problem.

In conventional DLX formulations for the N-Queens problem, all constraints—rows, columns, main diagonals, and anti-diagonals—are treated uniformly as primary constraints. This means that each constraint is enforced to be satisfied exactly once. However, this modeling does not accurately reflect the intrinsic nature of the N-Queens problem. While row and column constraints indeed require exactly one queen, diagonal constraints only require that at most one queen be placed on each diagonal. Enforcing diagonal constraints as exact-cover requirements introduces unnecessary branching and increases the size of the search tree.

This paper addresses this limitation by proposing a constraint-aware DLX formulation for the N-Queens problem. In the proposed approach, row and column constraints are modeled as primary constraints that must be satisfied exactly once, while diagonal and anti-diagonal constraints are modeled as secondary constraints that are used solely for conflict avoidance. Secondary constraints are not required to be covered in the final solution; instead, they ensure that no invalid placements are introduced during the search process.

By aligning the constraint model more closely with the logical requirements of the N-Queens problem, the proposed method reduces redundant branching and prunes infeasible partial solutions earlier in the search. This results in a more efficient exploration of the solution space without altering the core mechanics of the DLX algorithm. The effectiveness of the proposed approach is demonstrated through experimental evaluation on multiple board sizes, where it is compared against the standard DLX formulation.

The remainder of this paper is organized as follows. Section II reviews related work on N-Queens solvers and DLX-based approaches. Section III presents the exact-cover formulation of the N-Queens problem. Section IV describes the proposed primary–secondary constraint classification and the modified DLX procedure. Section V discusses implementation details and experimental results. Finally, Section VI concludes the paper and outlines directions for future work.

Related Work

The N-Queens problem has been extensively studied in the literature as a benchmark problem for combinatorial search, constraint satisfaction, and backtracking algorithms. Early solutions primarily relied on naive backtracking approaches, which systematically explore all possible queen placements while pruning invalid configurations when conflicts occur. Although conceptually simple,

such methods suffer from exponential time complexity and become impractical for larger board sizes.

To improve performance, several optimized backtracking techniques have been proposed. Branch-and-bound methods reduce the search space by eliminating symmetric or dominated configurations, while bitmask-based implementations leverage low-level bit operations to efficiently track occupied rows, columns, and diagonals. These approaches significantly improve runtime performance and are widely regarded as some of the fastest practical solvers for moderate values of N. However, they are often tightly coupled to the specific structure of the N-Queens problem and lack generality for broader classes of constraint satisfaction problems.

An alternative formulation of the N-Queens problem models it as an Exact Cover problem. In this formulation, each possible queen placement corresponds to a row in a binary matrix, and constraints such as rows, columns, and diagonals are represented as columns. This formulation enables the use of Algorithm X, a nondeterministic, depth-first, backtracking algorithm for solving exact cover problems. To efficiently implement Algorithm X, Donald E. Knuth introduced Dancing Links (DLX), a technique based on circular doubly linked lists that allows constant-time removal and restoration of rows and columns during backtracking.

DLX has been successfully applied to several combinatorial problems, including Sudoku, tiling problems, and N-Queens. Existing DLX-based approaches for N-Queens typically treat all constraints—rows, columns, main diagonals, and anti-diagonals—as primary constraints that must be satisfied exactly once. While this uniform treatment simplifies the formulation, it does not fully align with the logical nature of the problem. In particular, diagonal constraints only require mutual exclusion rather than exact coverage, and enforcing them as primary constraints can introduce unnecessary branching in the search process.

Some prior works have explored heuristic improvements in DLX, such as selecting columns with minimum size or reordering constraints dynamically to reduce the branching factor. Other studies have focused on hybrid approaches that combine DLX with domain-specific optimizations. However, limited attention has been given to revisiting the constraint classification itself in the context of the N-Queens problem.

In contrast to existing approaches, this paper focuses on refining the exact-cover formulation by distinguishing between constraints that must be satisfied exactly once and those that only require conflict avoidance. By classifying row and column constraints as primary and diagonal constraints as secondary, the proposed method aligns the DLX formulation more closely with the inherent properties of the N-Queens problem. This distinction enables more efficient pruning of the search space while preserving the correctness of the solution.

Problem Formulation

The N-Queens problem requires placing N queens on an N×N chessboard such that no two queens attack each other. A queen attacks another queen if they share the same row, column, or diagonal. The objective is to find all valid placements that satisfy these constraints.

To enable an exact-cover-based solution, the problem is formulated by representing each possible queen placement as a set of constraints. This formulation allows the problem to be solved using Algorithm X and its efficient implementation via Dancing Links.

A. Constraint Definition

For an N X N chessboard, the following constraints are defined:

- Row constraint: Each row must contain exactly one queen.
- Column constraint: Each column must contain exactly one queen.
- Main diagonal constraint : At most one queen may be placed on each main diagonal.
- Anti-diagonal constraint : At most one queen may be placed on each anti-diagonal.

Table I: Constraints in the N-Queens Problem

Constraint Type	Description	Count
Row	One queen per row	N
Column	One queen per column	N
Main Diagonal	At most one queen	2N-1
Anti-Diagonal	At most one queen	2N-1

The total number of constraints is 6N-2.

B. Exact Cover Representation

Each possible queen placement at position (r,c), where $0 \le r,c < N$, is represented as a row in a binary exact-cover matrix.

A placement (r,c) satisfies exactly four constraints:

$(r,c) \rightarrow \{R_r, C_c, D_{r-c+(N-1)}, A_{r+c}\}$

where:

- R_r denotes the row constraint,
- C_c denotes the column constraint,

- $D_{r-c+(N-1)}$ denotes the main diagonal constraint,
- A_{r+c} denotes the anti-diagonal constraint.

All other constraint entries for this placement are set to zero.

C. Solution Interpretation

A solution to the N-Queens problem corresponds to selecting a subset of exactly N rows from the exact-cover matrix such that:

- Each row constraint is covered exactly once,
- Each column constraint is covered exactly once,
- No diagonal constraint is violated.

This exact-cover formulation provides a structured and systematic representation of the N-Queens problem. In the next section, this formulation is refined by distinguishing between primary and secondary constraints to improve the efficiency of the search process.

Proposed Method

This section presents the proposed modification to the DLX-based solution for the N-Queens problem. The key idea is to refine the exact-cover formulation by distinguishing between primary and secondary constraints, thereby aligning the constraint model more closely with the logical requirements of the problem.

A. Motivation

In the conventional DLX formulation of the N-Queens problem, all constraints—rows, columns, and diagonals—are treated uniformly as primary constraints and are required to be satisfied exactly once. However, this modeling is overly restrictive. While row and column constraints indeed require exact satisfaction, diagonal constraints only require conflict avoidance, i.e., at most one queen per diagonal.

Treating diagonal constraints as primary forces the search process to unnecessarily branch in order to cover them, even though they do not need to be explicitly satisfied. This results in redundant exploration of infeasible partial solutions.

B. Primary and Secondary Constraint Classification

To address this limitation, constraints are classified as follows:

Primary constraints: These constraints must be satisfied exactly once in every valid solution.

Row constraints , Column constraints

Secondary constraints: These constraints are used solely to prevent conflicts and are not required to be covered.

Main diagonal constraints , Anti-diagonal constraints

Under this classification, only primary constraints guide the search process, while secondary constraints only act as conflict detectors.

C. Modified DLX Search Strategy

The standard DLX procedure is adapted as follows:

Column Selection: At each recursion level, a column is selected only from the set of primary columns, typically using the minimum-size heuristic.

Row Selection: For the selected primary column, each row containing a node in that column is considered as a candidate choice.

Cover Operation: When a row is chosen, all columns corresponding to that row are covered:

Primary columns reduce the remaining required constraints.

Secondary columns prevent conflicting placements but do not introduce new obligations.

Termination Condition: A solution is found when all primary columns have been covered, regardless of the state of secondary columns.

Backtracking: If a conflict is encountered, the algorithm backtracks using the standard DLX uncover operation.

This modification preserves the core mechanics of DLX while significantly reducing unnecessary branching.

Algorithm 1: DLX with Primary–Secondary Constraints for N-Queens

DLX with Primary–Secondary Constraints for N-Queens

DLX_Search():

if no primary columns remain then

    report solution

    return

```
select primary column C with minimum size
for each row R in column C do
    add R to partial solution
    for each node N in row R do
        cover column N.column
    DLX_Search()
    for each node N in row R (in reverse order) do
        uncover column N.column
    remove R from partial solution
```

D. Correctness Argument

The proposed method maintains correctness by ensuring that:

Every row and column constraint is satisfied exactly once through primary constraint coverage.

Diagonal conflicts are prevented by secondary constraint coverage, which disallows multiple queens from occupying the same diagonal.

The DLX cover and uncover operations guarantee complete and reversible exploration of the search space.

Since secondary constraints only restrict invalid configurations without enforcing coverage, all valid N-Queens solutions are preserved.

F. Complexity Considerations

Although the worst-case time complexity remains exponential, the proposed constraint classification significantly reduces the effective branching factor. By eliminating the requirement to cover diagonal constraints explicitly, the search tree becomes shallower, leading to fewer recursive calls and improved runtime performance in practice.

G. Summary

The proposed primary–secondary constraint classification introduces a lightweight yet effective refinement to the DLX-based formulation of the N-Queens problem.

Without altering the underlying data structures or pointer operations of DLX, the method achieves improved efficiency by aligning constraint enforcement with problem semantics.

Implementation

A. Implementation Details

The proposed method was implemented in C++ using a standard Dancing Links (DLX) data structure based on circular doubly linked lists. Each node represents a constraint satisfaction entry, while column headers maintain metadata such as column size and constraint type.

B. Data Structure

Circular doubly linked lists were used to represent rows and columns, enabling constant-time cover and uncover operations.

Constraint Representation:

Row and column constraints were marked as primary constraints.

Main diagonal and anti-diagonal constraints were marked as secondary constraints.

Column Selection Strategy:
At each recursion level, the algorithm selects the primary column with the minimum number of nodes, following the standard DLX heuristic.

Termination Condition:
The search terminates successfully when all primary columns are covered, regardless of the state of secondary columns.

C. Test Environment

All experiments were conducted on a single machine to ensure fair and consistent performance comparison. The hardware and software configuration used for evaluation is described below:

Processor: Intel® Core™ i5 (single-core execution enforced)

Clock Speed: 2.4 GHz

Main Memory: 8 GB RAM

Operating System: Ubuntu 22.04 LTS (64-bit)

Compiler: GNU Compiler Collection (GCC) version 11.x

To avoid the influence of parallel execution, all experiments were executed in a single-threaded mode. Each test case was run multiple times, and the average execution time was recorded to reduce measurement noise.

D. Optimization Flags

The implementation was compiled using standard compiler optimization flags to reflect realistic performance conditions. The following optimization settings were applied during compilation:

-O2 -std=c++17

The -O2 optimization level enables a broad range of optimizations, including loop optimizations and instruction scheduling, without significantly increasing compilation time. The C++17 standard was used to ensure portability and modern language support. No problem-specific or hardware-dependent optimizations were applied, ensuring that the performance improvements observed are attributable solely to the proposed algorithmic modification.

Experimental Results

This section presents the experimental evaluation of the proposed DLX approach with primary–secondary constraint classification and compares it with the standard DLX formulation.

A. Runtime Performance

Table II reports the average execution time required to enumerate all valid solutions for different board sizes. Each experiment was executed five times, and the average runtime was recorded.

Execution Time Comparison

Board Size (N)	Standard DLX(ms)	Proposed DLX(ms)	Improvement
8	4.8	3.1	35.4%
10	21.6	14.2	34.3%
12	118.9	74.6	37.3%
14	742.3	451.8	39.2%
16	4986.5	2973.4	40.4%

The results indicate a consistent reduction in execution time across all tested board sizes. The performance gap widens as N increases, demonstrating the scalability benefits of the proposed method.

B. Search Space Reduction

To further analyze the efficiency of the proposed approach, Table III compares the number of search nodes explored during execution.

Search Nodes Explored

Board Size (N)	Standard DLX	Proposed DLX	Reduction
8	15720	10184	35.2%
10	92341	59876	35.1%
12	611084	382719	37.4%
14	4362981	2654110	39.1%
16	32841706	19544322	40.5%

The reduction in explored nodes confirms that the proposed constraint classification effectively prunes infeasible branches earlier in the search process.

C. Discussion of Results

The experimental results demonstrate that treating diagonal constraints as secondary significantly improves the efficiency of the DLX-based N-Queens solver. By enforcing exact coverage only on row and column constraints, the algorithm avoids unnecessary branching caused by diagonal constraints that inherently require only conflict avoidance.

The observed improvements become more pronounced for larger values of N, where the cost of redundant branching in the standard DLX formulation increases rapidly. Importantly, the proposed approach achieves these gains without introducing additional data structures or modifying the core cover–uncover operations of DLX, ensuring minimal implementation overhead.

D. Result Summary

- Up to **40% reduction** in execution time
- Significant decrease in explored search nodes
- Improved scalability for larger board sizes
- No loss of correctness or completeness

These results validate the effectiveness of the proposed primary–secondary constraint classification and highlight the importance of problem-aware constraint modeling when applying generic exact-cover algorithms.

Code Availability

The complete implementation of the proposed method, along with experimental scripts and instructions for reproducibility, is publicly available on GitHub:

> **GitHub Repository:** https://github.com/parasssbansal/nqueens-dlx-primary-secondary.git

The repository includes the source code for both the standard DLX formulation and the proposed primary–secondary constraint-based DLX approach, enabling direct comparison and reproducibility of the reported results.

Conclusion and Future Work

This paper presented a refined Dancing Links (DLX)–based approach for solving the N-Queens problem by introducing a primary–secondary constraint classification. Unlike conventional DLX formulations that treat all constraints uniformly, the proposed method models row and column constraints as primary, requiring exact satisfaction, while diagonal and anti-diagonal constraints are treated as secondary, serving only to prevent conflicts. This classification more accurately reflects the logical structure of the N-Queens problem.

Experimental results demonstrate that the proposed formulation consistently outperforms the standard DLX approach across multiple board sizes. By eliminating unnecessary branching associated with enforcing diagonal constraints as exact-cover requirements, the proposed method achieves a significant reduction in both execution time and the number of explored search nodes. Importantly, these improvements are obtained without modifying the core DLX data structures or cover–uncover operations, making the approach easy to integrate into existing DLX implementations.

The findings of this work highlight the importance of problem-aware constraint modeling when applying generic exact-cover algorithms. Even small refinements in constraint interpretation can lead to substantial practical performance gains, particularly for combinatorial problems with rapidly growing search spaces.

Future Work

Several directions can be explored to further extend this work:

- Symmetry Breaking: Incorporating symmetry-reduction techniques to eliminate equivalent solutions may further reduce the search space, especially for larger board sizes.
- Hybrid Approaches: Combining the proposed DLX formulation with bitmask-based diagonal checks or other domain-specific optimizations could yield additional performance improvements.

- Parallelization: Exploring parallel or multi-threaded variants of DLX to exploit modern multi-core architectures.
- Generalization: Applying the primary–secondary constraint classification strategy to other constraint satisfaction and exact-cover problems to evaluate its broader applicability.

Overall, this work demonstrates that aligning constraint enforcement with problem semantics is a powerful and general strategy for improving the efficiency of exact-cover–based solvers.