BDD-Based Optimization for the Quadratic Stable Set Problem

Jaime E. González[a], Andre A. Cire[b], Andrea Lodi[a], Louis-Martin Rousseau[a]

[a]*Department of Mathematics and Industrial Engineering, Polytechnique Montréal*
{jaime.gonzalez,andrea.lodi,louis-martin.rousseau}@polymtl.ca
[b]*Dept. of Management, University of Toronto Scarborough and Rotman School of Management*
andre.cire@rotman.utoronto.ca

Abstract

The quadratic stable set problem (QSSP) is a natural extension of the well-known maximum stable set problem. The QSSP is NP-hard and can be formulated as a binary quadratic program, which makes it an interesting case study to be tackled from different optimization paradigms. In this paper, we propose a novel representation for the QSSP through binary decision diagrams (BDDs) and adapt a hybrid optimization approach which integrates BDDs and mixed-integer programming (MIP) for solving the QSSP. The exact framework highlights the modeling flexibility offered through decision diagrams to handle nonlinear problems. In addition, the hybrid approach leverages two different representations by exploring, in a complementary way, the solution space with BDD and MIP technologies. Machine learning then becomes a valuable component within the method to guide the search mechanisms. In the numerical experiments, the hybrid approach shows to be superior, by at least one order of magnitude, than two leading commercial MIP solvers with quadratic programming capabilities and a semidefinite-based branch-and-bound solver.

Keywords: Decision Diagrams, Hybrid Optimization, Quadratic Stable Set Problem, Binary Quadratic Programs, Dynamic Programming

1. Introduction

A stable set is a pairwise non-adjacent subset of vertices in a graph and defines a fundamental structure in discrete optimization. Classical problems associated with stable sets, such as the *maximum stable* (or *independent*) *set problem*, have been extensively studied in the optimization literature and arise in applications such as social networks [1], data mining [2] and computational biology [3], to name a few. We refer to Wu and Hao [4] for a survey of existing methodologies and other applications.

Of growing interest in the optimization community is the *quadratic stable set problem* (QSSP), a difficult variant of the maximum stable set problem where additional profits are associated with pairs of vertices. The QSSP is a key component of modern applications (e.g., protein structure prediction [5] and marketing [6]), but its related computational methodologies are still limited in comparison to classical stable set problems. Moreover, given the natural relationship between stable sets and cliques (i.e., induced complete subgraphs) in a graph, the study of the QSSP can contribute to approaches for solving clique related problems. To the best of our knowledge, the QSSP first appeared as a subproblem when estimating the quality of cellular networks [7], later addressed through mixed-integer programming (MIP) reformulations [8]. Karimi and Ronagh [9] also investigate Lagrangian methods and report experiments for instances of up to 30 vertices.

The QSSP can also be formulated as a binary quadratic problem (BQP) and addressed via non-linear solvers. In this context, Furini and Traversi [10] develop a semidefinite programming (SDP) relaxation that is used as a bounding mechanism in a branch-and-bound search, solving instances with up to 100 vertices. In a related approach, another generic methodology for solving the QSSP is BiqCrunch [11], a state-of-the-art semidefinite-based solver which has solved instances having up to 150 vertices. The QSSP was also used as a benchmark in a computational study of different linearization techniques for BQPs [12] using a MIP solver, demonstrating that current generic methodologies are still limited to graphs with 150 vertices. Nevertheless, leading commercial MIP solvers incorporate flexible quadratic programming (QP) capabilities that allow to directly tackle the BQP formulation of the QSSP through nonlinear programming (NLP) based branch and bound. Such an alternative is worth being considered as another benchmark for this problem.

In this paper, we propose a novel QSSP solution approach based on *decision diagrams* (DDs). The theory and practice of DDs for optimization has been gradually established as a fruitful research area in operations research [13]. A variety of stand-alone methodologies, decomposition approaches, and integrated techniques based on decision diagrams contributed to novel state-of-the-art methods in a large array of applications [14, 15, 16, 17, 18, 19]. In particular, decision diagrams have been effective for non-linear integer optimization problems [20, 21] as they provide a *discrete* type of relaxation which can be leveraged as an alternative bounding mechanism.

Our exact methodology exploits the strength of both DD-based relaxations and QP capabilities in MIP technology to solve the QSSP more efficiently. Specifically, we model and solve the QSSP via a hybrid approach which integrates binary decision diagrams (BDDs), MIP, and machine learning, following the generic framework proposed in [15]. The hybrid BDD-MIP algorithm is based on a BDD-based search mechanism that branches on underlying equivalent classes of variable assignments, here represented as states in a dynamic programming reformulation of the problem. We describe a novel BDD representation for the QSSP and highlight the flexibility of the decision diagram modeling framework to deal with quadratic problems. We also remark how the hybrid approach leverages two different representations of a problem in a collaborative framework.

Moreover, supervised learning plays an important role within the hybrid approach to guide the exploration of the solution space. In our algorithm, we use traditional machine learning to train a classifier which dynamically selects whether a branch should be explored either by MIP or BDD technology. We present computational experiments using the proposed BDD-MIP algorithm and compare its performance against general-purpose approaches; namely, two leading commercial MIP solvers with QP capabilities and, a competitive semidefinite-based solver. Numerical results show that the BDD-MIP can significantly outperform such solvers in existing benchmarks.

The remainder of the paper is organized as follows. Section 2 defines the quadratic stable set problem and introduces relevant notation. Section 3 presents the decision diagram representation for the problem, which relies on a novel dynamic programming reformulation of the problem. Section 4 describes the hybrid BDD-MIP optimization approach adapted for the QSSP. Finally, the experimental evaluation is presented in Section 5, while Section 6 contains concluding remarks.

2. The Quadratic Stable Set Problem

Let $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V} := \{1, \ldots, n\}$ is a set of n vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges. A stable set of $\mathcal{G}$ is a subset of vertices $\mathcal{S} \subseteq \mathcal{V}$ where no two vertices are connected by an edge in $\mathcal{E}$. With each vertex $i \in \mathcal{V}$ we associate a profit w_i that is collected if vertex i is included in $\mathcal{S}$. Moreover, we consider a profit q_{ij} for each pair of vertices $i, j \in \mathcal{V}$ that is collected if both vertices are included in $\mathcal{S}$. We denote by $Q = \{q_{ij}\}_{i,j=1,\ldots,n} \in \mathbb{R}^{n \times n}$ the profit matrix, here not restricted to be positive semidefinite.

The quadratic stable set problem (QSSP) asks for the stable set in $\mathcal{G}$ with maximum total profit. For a mathematical formulation, let us define x_i as a binary variable that takes value of 1 if vertex $i \in \mathcal{V}$ belongs to the stable set $\mathcal{S}$ and 0 otherwise. The QSSP can be formulated as the following binary quadratic program (BQP):

$$\max_x \quad \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2 q_{ij} x_i x_j \tag{1}$$

$$\text{subject to} \quad x_i + x_j \leq 1, \qquad \forall \{i, j\} \in \mathcal{E}, \tag{2}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in \mathcal{V}. \tag{3}$$

Note that $x_i^2 = x_i$ for all $i \in \mathcal{V}$ then the linear profits w_i can be easily adjusted and consider the terms q_{ii} in Q, i.e., its main diagonal. An alternative formulation for the QSSP is obtained by leveraging the concept of a *clique cover*, i.e., a partition of the vertices of $\mathcal{G}$ into cliques. Namely, let $\mathcal{K}$ be any collection of cliques that covers $\mathcal{G}$. Since each stable set can contain at most one vertex in a clique $K \in \mathcal{K}$, the *clique formulation* for the QSSP is obtained by replacing inequalities (2) with:

$$\sum_{i \in K} x_i \leq 1, \qquad\qquad \forall K \in \mathcal{K}, \tag{4}$$

which is well-known to provide stronger relaxations when only linear costs are considered [22]. The set $\mathcal{K}$ is typically constructed using maximal cliques that can be computed efficiently, e.g., by a greedy procedure [13]. In particular, a maximal clique K is obtained by initially selecting the vertex with highest degree. Next, we iteratively include adjacent vertices (sorted by highest degree) to each vertex in K until no more inclusions are possible. We then add clique K to set $\mathcal{K}$ and remove from $\mathcal{G}$ all the edges belonging to the included clique. We update the vertices degrees, and repeat the procedure.

As an illustrative example, consider the QSSP instance defined by an undirected graph with 5 vertices and 5 edges presented in Figure 1. Red numbers next to each vertex $i \in \mathcal{V}$ correspond to the vertex profit w_i. We also provide the matrix of quadratic profits Q. In the figure, vertices 1, 2 and 5 form a stable set whose total profit is -3 yielded by $w_1 + w_2 + w_5 + 2 \cdot q_{12} + 2 \cdot q_{15} + 2 \cdot q_{25}$. The optimal stable set for the illustrative example consists of vertices 1 and 2 with optimal objective value of 9.

$$Q = \begin{bmatrix} 0 & 3 & 2 & 0 & -4 \\ 3 & 0 & -2 & -5 & -3 \\ 2 & -2 & 0 & 0 & 1 \\ 0 & -5 & 0 & 0 & 2 \\ -4 & -3 & 1 & 2 & 0 \end{bmatrix}$$

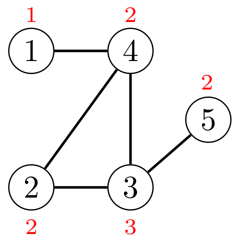


Figure 1: Undirected graph and matrix Q for an illustrative QSSP example

For notation purposes, we denote by z^* the optimal objective value of the QSSP. We also note in passing that the problem reduces to the classical maximum weighted stable set problem either when $Q = \mathbf{0}$, or when the quadratic profits are non-positive, i.e., $Q \leq \mathbf{0}$, the latter case requiring a non-trivial transformation of the original graph [7]. In addition, we observe that when only quadratic profits are considered (i.e., $w_i = 0,\ i \in \mathcal{V}$), the resulting particular QSSP could be reformulated as a *maximum edge-weighted clique problem* [23, 24, 25, 26] in the complement graph $\bar{\mathcal{G}}$ where profits q_{ij} become the weights on edges $(i, j) \in \bar{\mathcal{E}}$. Nevertheless, since the maximum stable set problem is NP-hard in the strong sense [27], the same is true for the QSSP, which in turn is generally more computationally challenging than the classical linear version.

3. A Decision Diagram Representation for the QSSP

Conceptually, a decision diagram is a compressed representation of the state-transition graph of a dynamic programming (DP) model [13]. In this section, we begin by deriving a novel problem representation for the QSSP through a DP formulation (Section 3.1). Next, we describe how to extract the resulting decision diagram representation from such formulation (Section 3.2), which will be central to our exact methodology.

3.1. A dynamic programming model for the QSSP

Before introducing our DP formulation to QSSP, we first reformulate (1)-(3) in order to reveal recursive structure. Namely, note that the quadratic model

$$\begin{aligned}
\max_{x,s} \quad & \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} x_i s_i && && (5)\\
\text{subject to} \quad & x_i + x_j \leq 1, && \forall \{i,j\} \in \mathcal{E}, && (6)\\
& s_i = \sum_{j=i+1}^{n} 2q_{ij} x_j, && \forall i \in \mathcal{V}, && (7)\\
& x_i \in \{0,1\}, && \forall i \in \mathcal{V}. && (8)
\end{aligned}$$

is also valid to QSSP, where the difference with respect to the original model is the introduction of variables s_i representing the inner sum within each quadratic term i of the objective function (1).

Using model (5)-(8), we can now evaluate the marginal impact of adding a vertex $i \in \mathcal{V}$ to a given stable set as follows. Suppose that variables $x_1, \ldots, x_{i-1}$ are fixed to values that can be extended to a feasible solution (e.g., by appending variables $x_i, \ldots, x_n$ and setting them to zero), thereby defining a stable set $S := \{k \in \{1, \ldots, i-1\} \colon x_k = 1\}$. This, in turn, leads to the *eligible* set

$$\mathcal{I} := \{j \in \mathcal{V} \colon \{j,k\} \notin \mathcal{E} \ \text{ for all } k \in S\}$$

of all vertices that can still be added to the stable set S; i.e., vertex i can be added to S (i.e., x_i set to one) only if $i \in \mathcal{I}$. Furthermore, if $i \geq 2$ is added to S, we collect a profit $w_i + s_i$ (and only a profit of w_i if $i = 1$). Thus, to evaluate whether i can be added to a (partial) stable set and the resulting profit, it suffices to have the set $\mathcal{I}$ of eligible vertices and the partial sum s_i as defined in (7).

This leads to the following DP reformulation of QSSP. Given a fixed vertex ordering $1, \ldots, n$, we consider a system of $n+1$ stages where, at each stage i, we decide whether to add or not a vertex $i \in \mathcal{V}$, as represented by the value of variable $x_i \in \{0,1\}$. The state of the system at each stage i is a pair $(\mathcal{I}, s)$, where $\mathcal{I} \subseteq \mathcal{V}$ is the set of eligible vertices according to the assignment in stages

$1, \ldots, i-1$ and $s = (s_i, s_{i+1}, \ldots, s_n)$ is a vector of $n-i+1$ elements with the sums (7) indexed from i to n. We remark that the definition of state s is related to the state information defined for unconstrained binary quadratic programming in [28]. When assigning x_i (i.e., deciding whether to include i or not), we transition to a new state defined by the function

$$g_i(\mathcal{I}, s, x_i) := \begin{cases} (\mathcal{I} \setminus N_i, (s_{i+1} + 2q_{i+1\,i}, \,\ldots,\, s_n + 2q_{ni})), & \text{if } x_i = 1, \\ (\mathcal{I}, (s_{i+1}, \,\ldots,\, s_n)), & \text{otherwise.} \end{cases}$$

where $N_i = \{i\} \cup \{j \in \mathcal{V} \colon \{i,j\} \in \mathcal{E}\}$ is the neighborhood of i including the vertex itself. That is, we update the eligible set $\mathcal{I}$ according to x_i and the sum state s according to (7). The total profit in terms of x_i is

$$h_i(\mathcal{I}, s, x_i) := w_i x_i + \mathbb{I}(i \geq 2) s_i x_i,$$

where $\mathbb{I}(C)$ is an indicator function that evaluates to 1 if condition C is true and 0 otherwise. Finally, a vertex i can only be added if it belongs to the eligibility set $\mathcal{I}$, i.e., the set of feasible assignments at a stage i is

$$\mathcal{F}_i(\mathcal{I}) := \{0\} \cup \{\, \mathbb{I}(i \in \mathcal{I}) \,\}.$$

Equipped with $\mathcal{F}_i(\cdot)$, the transition function $g_i(\cdot)$, and the profit function $h_i(\cdot)$, an optimal solution x^* to QSSP solves the Bellman equations

$$V_i((\mathcal{I}, s)) = \max_{x_i \in \mathcal{F}_i(\mathcal{I})} \left\{ h_i(\mathcal{I}, s, x_i) + V_{i+1}(g_i(\mathcal{I}, s, x_i)) \right\}, \quad i = 1, \ldots, n, \tag{9}$$

$$V_{n+1}((\mathcal{I}, s)) = 0, \tag{10}$$

where $V_1((\mathcal{V}, \mathbf{0}))$ yields the optimal solution value of the QSSP. In particular, we denote $(\mathcal{V}, \mathbf{0})$ by *root state* of the system.

3.2. Constructing the BDD representation for the QSSP

We now describe how to generate the decision diagram representation based on the dynamic program presented above. For notation purposes, let $\mathcal{S}(\mathcal{G})$ be the family of stable sets of $\mathcal{G}$.

A binary decision diagram for a QSSP instance is a layered directed acyclic graph $\mathcal{B} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ is the node set and $\mathcal{A}$ is the arc set. The node set $\mathcal{N}$ is partitioned into $n+1$ layers $L_1, \ldots, L_{n+1}$, where the first and last layers L_1 and L_{n+1} are singletons containing a *root node* r and a *terminal node* t, respectively. We denote by $l(u)$ the index of the layer of a node $u \in \mathcal{N}$, i.e., $u \in L_{l(u)}$. A BDD arc $a \in \mathcal{A}$ only connects nodes in adjacent layers and is equipped with a binary label $d_a \in \{0, 1\}$ and a profit h_a. We refer to arc a by *1-arc* if $d(a) = 1$ and by *0-arc* otherwise.

A BDD is a compact graphical representation of the state transition graph of the DP (9)-(10). Specifically, each node in $\mathcal{N}$ represents a state $(\mathcal{I}, s)$ and the layer L_i contains the nodes associated with the states that are reachable at stage i, $i = 1, \ldots, n$. In particular, the root node r is associated with the root state $(\mathcal{V}, \mathbf{0})$. Arcs encode the transition $g_i(\cdot)$, i.e., there exists an arc $a = (u, u') \in \mathcal{A}$ with label d_a iff, given the state $(\mathcal{I}, s)$ associated with u, we have $d_a \in \mathcal{F}(\mathcal{I})$ and the state associated with u' is $g_{l(u)}(\mathcal{I}, s, d_a)$. The profit of such arc is $h_a := h_{l(u)}(\mathcal{I}, s, d_a)$. The terminal node t, in turn, represents all terminal states of (9)-(10), i.e., they are perceived as merged into a single node.

In such a representation, we have a one-to-one mapping between stable sets in $\mathcal{G}$ and paths of the BDD. Namely, for every arc-specified path $(a_1, a_2, \ldots, a_n)$ starting from the root r and ending at the terminal t, the arc labels yield a feasible assignment $x := (d_{a_1}, \ldots, d_{a_n})$ by validity of the DP. Conversely, every such feasible assignment must be encoded in some path of the BDD. This implies that the QSSP now reduces to finding a longest-path problem over the BDD, where arc lengths are the arc profits h_a.

Figure 2 illustrates the exact BDD for the instance in Figure 1. States are included on top of each of their corresponding nodes in each layer. Solid and dash arcs represent $d(a) = 1$ and $d(a) = 0$, respectively, and arc profits h_a are included (in blue) on top of each arc.

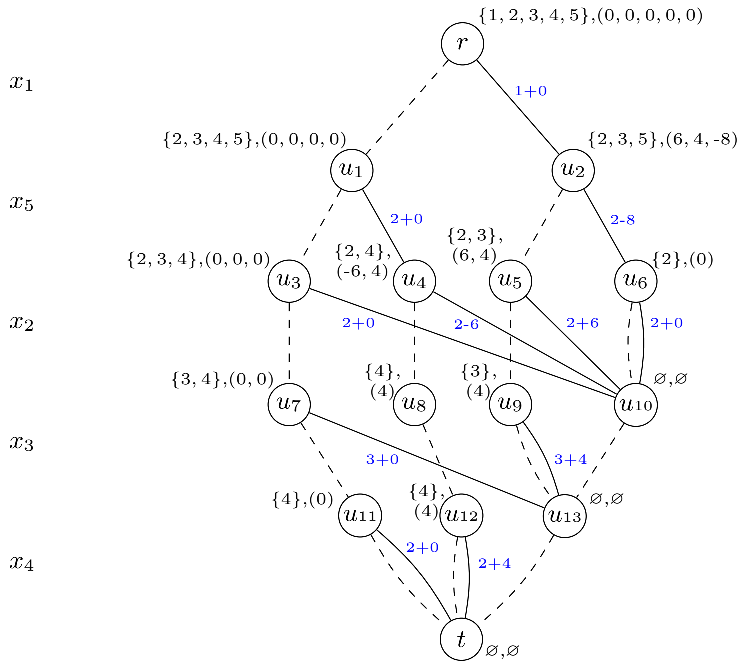


Figure 2: Exact BDD for the QSSP instance of Figure 1

The variable ordering in the decision diagram of Figure 2 is established by x_1, x_5, x_2, x_3, x_4, i.e., these are the variables associated with outgoing arcs from layers L_1,L_2, L_3,L_4,L_5, respectively. To illustrate the BDD compilation, we observe that the root node r has state information $(\{1, 2, 3, 4, 5\}, (0, 0, 0, 0, 0))$. Note that, at the beginning, all vertices are candidate to be added to the stable set and, no matter the vertex associated with outgoing arcs of layer L_1 only the vertex weight w_i is collected. Next, for instance, there is an outgoing 1-arc which represents $x_1 = 1$ and leads to node u_2 with state information $(\mathcal{I}(u_2), s(u_2)) = (\{2, 3, 5\}, (6, 4, \text{-}8))$. The latter indicates that by including vertex 1 in the partial stable set, we get to a state where we remove vertex 1 and 4 from the possible vertices, updating states accordingly.

In addition, note that variable x_5 is associated with L_2. Consequently, the weight of the outgoing 1-arc (u_2, u_6) is $2 - 8$, this is, the profit of vertex 5 (i.e., $w_5 = 2$) plus "-8" which is associated with the sum of quadratic contributions for vertex 5 in the state component $s(u_2)$.

In the BDD in Figure 2, the longest path is given by the arc-specified path $((r, u_2),(u_2, u_5),(u_5, u_{10}),(u_{10},u_{13}),(u_{13}, t))$ indicating that the optimal stable set for the problem is $\{1, 2\}$ with optimal objective value $z^* = 9$.

Compiling BDDs. We follow a typical top-down procedure for constructing an exact BDD, which is equivalent to a forward recursion over the DP. The procedure is defined as follows.

Let us denote by $(\mathcal{I}(u), s(u))$ the state associated with the BDD node $u \in \mathcal{N}$. Layers are compiled one at a time in the order $L_1, \ldots, L_{n+1}$. At each iteration $i = 1, \ldots, n$, we calculate all state transitions from the states associated with nodes in L_i to generate nodes in L_{i+1}, adding arcs as necessary. We also ensure that no two nodes have the same state, i.e., either $\mathcal{I}(u) \neq \mathcal{I}(u')$ or $s(u) \neq s(u')$ for any two nodes u, u' at the same layer. Finally, all nodes in the last layer L_{n+1} are merged into a single terminal node t.

3.3. Approximate decision diagrams for the QSSP

In general, exact BDDs grow exponentially large on the problem input and are not computationally tractable. Because of this, we instead manipulate the so-called *approximate* versions, i.e., relaxed and restricted decision diagrams. Such diagrams are key as a bounding mechanism since they exploit discrete structure to relax the state space.

A DD is *relaxed* if it over-approximates the solution set of a problem, encoding all feasible solutions but also allowing infeasible ones. Relaxed BDDs are obtained similarly as exact BDDs when using a top-down approach. In particular, if the number of nodes in a layer (i.e., the layer width) exceeds a given pre-specified limit W during its construction, two non-identical nodes u and u' are heuristically selected and merged into a new node u''. Next, the longest (resp., shortest) path value in a relaxed BDD yields a dual bound for a maximization (resp., minimization) optimization problem. Note that the maximum width W is then a relevant parameter because it allows to trade-off computational effort and bound quality, i.e., the larger the W value, the better the bound that is obtained. More details on the experiments leading to a suitable value of W are reported in Section 5.

For the QSSP, we propose a valid merging operator which guarantees that no feasible solution is lost meanwhile keeping the relaxed BDD size under control. The state of the merged node u'' is set as the pair $(\mathcal{I}(u''), s(u''))$ where $\mathcal{I}(u'') := \mathcal{I}(u) \cup \mathcal{I}(u')$ and $s(u'') := \max\{\{s(u)\}_j, \{s(u')\}_j\}_{j \in \mathcal{I}(u) \cup \mathcal{I}(u')}$. The strategy used to define which nodes are merged consists on selecting BDD nodes u and u' with the partial longest path from the root node. Once nodes are merged, all previous incoming arcs to u and u' are directed to the new merged node u''. The proposed merging operator assures that all valid stable sets are preserved and that a longest path computation in the resulting relaxed BDD provides an upper bound on the optimal objective value z^*.

Furthermore, we also manipulate *restricted* decision diagrams to obtain feasible solutions. A restricted BDD under-approximates the solution set of a problem i.e., it only allows feasible solutions but could miss the optimal one. They can be also compiled through a top-down construction. In the restricted-version case, when reaching the maximum width W in a layer, instead of merging nodes, we heuristically select nodes to be removed from such a layer. A longest path computation in this case provides a lower bound on z^*.

4. A BDD-based Hybrid Optimization Approach for the QSSP

Given relaxed and restricted BDD representations as well as a BQP formulation of the QSSP (presented in Sections 2 and 3), we propose to deploy a hybrid BDD-MIP ([15]) algorithm as a solution methodology. The integrated BDD-MIP method leverages two problem representations and BDD-based search mechanisms to exploit complementary strengths coming from the different optimization paradigms. In Section 4.1, we initially describe a typical BDD-based exploration of the solution space. Next, we focus on the hybrid algorithm mechanisms considered for the QSSP (Section 4.2).

4.1. BDD-based search scheme

A key component of the combined framework is the BDD-based exploration in which a relaxed binary decision diagram plays the role of a search tree in a branch-and-bound scheme. In such a search mechanism, the solution space is divided and explored by recursively branching on suitable BDD nodes (i.e., set of partial solutions) instead of branching on variable-value pairs as in traditional linear programming-based branch and bound.

We now describe the general idea of a BDD-based search mechanism in the context of a stand-alone decision diagram approach [14]. Consider a relaxed binary decision diagram $\bar{\mathcal{B}}$ of the QSSP as the example provided in Figure 3, for the illustrative instance in Figure 1, where the maximum width is set as $W = 2$. For every pair of nodes $u, u' \in \bar{\mathcal{B}}$ such that $l(u) < l(u')$, let $\bar{\mathcal{B}}_{uu'}$ be the binary decision diagram induced by all the nodes and arcs that lie on directed paths from u to u', e.g., $\bar{\mathcal{B}}_{rt} = \bar{\mathcal{B}}$. We say that a node u in $\bar{\mathcal{B}}$ is *exact* if all $r - u$ paths lead to the same state $s(u)$, and is relaxed otherwise. In addition, a *cutset* of $\bar{\mathcal{B}}$ is a subset of nodes C such that any $r - t$ path in $\bar{\mathcal{B}}$ contains at least one node in C. In specific, C defines an exact cutset if all nodes in C are exact. Several strategies have been proposed for obtaining exact cutsets, such as *the frontier cutset (FC)* and *the last exact layer (LEL)* (we refer the reader to [13]).

Figure 3 illustrates an exact cutset C defined by the LEL and formed, in this case, by BDD nodes $\bar{u}_1$ and $\bar{u}_2$ (in orange). Note that nodes in blue, $\bar{u}_3$ and $\bar{u}_4$, were forcefully merged to meet W using the merging operator proposed in Section 3.3.

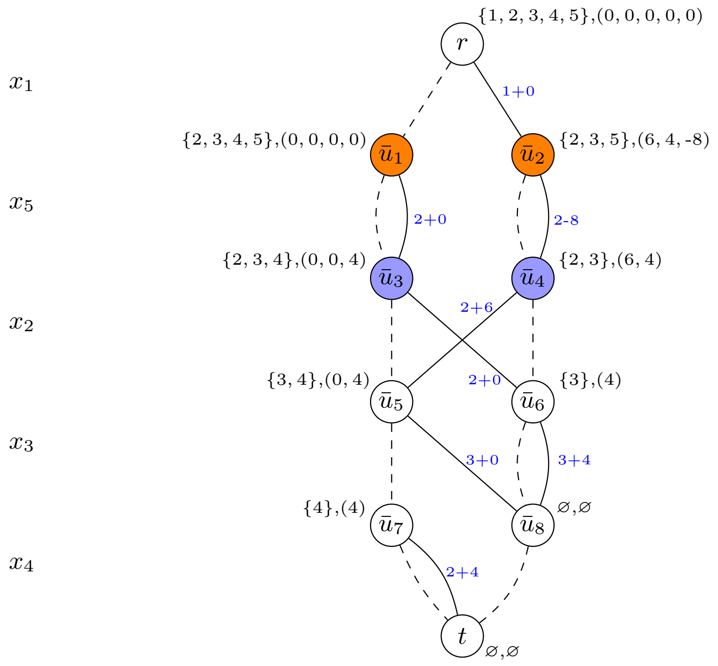


Figure 3: Relaxed BDD for the QSSP instance of Figure 1

Given an exact cutset C, a BDD-based branch and bound explores each BDD node in C to find and prove the optimal solution. Let $v^*(u)$ be the longest-path value from r to u for each $u \in C$. Let z_u^* be the optimal value of the subproblem for which its solutions are exactly encoded in $\mathcal{B}_{ut}$, therefore $v^*(u) + z_u^*$ is the value of the best solution across all $r - t$ paths that contain BDD node u. Since all $r - t$ paths of a decision diagram must contain some node in C, we search the optimal value z^* of the problem by solving the subproblems associated with $\mathcal{B}_{ut}$. Such subproblems (every $u \in C$) are solved separately and each subproblem then leads to a smaller and hence more tractable binary decision diagram. This procedure can be applied recursively for each u if the relaxed BDD rooted in u is either not exact (i.e., if the maximum width is reached while its construction) or cannot be implicitly pruned.

As an illustration of the BDD-based exploration scheme in Figure 3, we have to explore BDD nodes $\bar{u}_1$ and $\bar{u}_2$ further to continue searching for the optimal solution. In a stand-alone BDD-based search, such an exploration implies the compilation of two different relaxed decision diagrams, each rooted in $\bar{u}_1$ and $\bar{u}_2$, respectively.

4.2. Hybrid BDD-MIP mechanisms

In the integrated approach, MIP technology is incorporated into the BDD-based exploration scheme. Namely, a MIP solver can directly prune BDD nodes by solving them up to optimality while also providing lower bounds. Meanwhile, the incumbent solution found so far in the BDD-based branch and bound can be used to define a lower cutoff in the objective function of the subproblems explored by the MIP solver.

4.2.1. MIP-based pruning strategy

We specifically propose to adapt the *IP-based pruning* strategy in [15]. Consider a BDD-based branch and bound and a relaxed BDD which has to be explored further. Once a cutset C is defined and selected, a node $u \in C$ can be directly explored and pruned in advance by finding its optimal solution z^*_u. Along these lines, the strategy consists of solving the subproblem associated with u using a MIP solver as opposed to recursively relax and explore $\mathcal{B}_{ut}$ through BDD technology. In addition, once the subproblem associated with node u is solved, $v^*(u) + z^*_u$ also establishes a lower bound on z^*.

For the QSSP, a subproblem encoded in a BDD node u corresponds to a vertex-induced subgraph defined by the vertices considered in the state component $\mathcal{I}(u)$. Note that such a subproblem also leads to a BQP model if it is tackled with MIP technology. Thus, the mechanisms within the hybrid BDD-MIP algorithm leads to an algorithm-selection decision during the exploration. There are three important aspects to be considered, (i) the subproblems (i.e., BDD nodes to be explored) are dynamically generated during search, (ii) solving a BQP subproblem up to optimality may be computationally too expensive, and (iii) such an algorithm-selection decision is made based on the subproblem features encoded by the state $(\mathcal{I}(u), s(u))$. Determining whether to apply the MIP-based pruning at each BDD node plays a central role in the hybrid approach, and we cast this algorithmic question as a classification task.

In [15], when solving the maximum stable set problem, machine learning (ML) is used to derive trained classifiers and a decision tree to determine when to use the MIP-based pruning strategy. At the root node, a classifier automatically detects whether the related subproblem (i.e., the instance itself) is simply solved by MIP technology. Then, at the remaining BDD nodes, a computationally cheap decision tree determines if each BDD node is either pruned using a MIP solver or the BDD continues the exploration further. In this paper, we take a further step and propose using ML to guide the exploration within the hybrid BDD-MIP algorithm but at every BDD node subproblem.

4.2.2. ML-driven exploration

We rely on the recent connection between ML and discrete optimization (see, e.g., [29] for a survey). Specifically, we cast the algorithm-selection decision within the hybrid BDD-MIP algorithm as a classification task which is addressed by ML. We employ traditional supervised ML techniques and learn a classifier to decide which technology should be used (i.e., a BDD relaxation or a MIP representation) to explore, on-the-fly, a BDD subproblem.

In a supervised classification problem, the objective is to learn a function which assigns a discrete class label to an unseen instance, given a set of already classified examples (i.e., the training data). Each example in the training set is described by a set of features (attributes) and an associated label. The learned function (classifier) then maps the features to the available classes revealing a possible hidden structure of the training dataset. In our case, an example corresponds to a QSSP instance and we define two class labels that represent the BDD and MIP technology.

We perform the learning experiments offline to train the classifier which is later incorporated in the hybrid BDD-MIP algorithm. Thus, we proceed to describe the supervised learning methodology and the experiments to learn such a classifier. The methodology comprises five main steps, (i) the generation of instances, (ii) the feature design, (iii) the label definition for the learning task, (iv) the dataset composition, and finally, (v) the learning experiments.

Instances generation. As previously mentioned, a QSSP instance mainly corresponds to both an undirected graph and a symmetric matrix of quadratic profits. We randomly produce examples for the training dataset by generating graphs following the Erdös-Rényi (ER) [30] model where we vary the number of nodes (n) and graph density (p). Moreover, for each instance, we generate a symmetric matrix Q where we define its percentage of positive coefficients (v).

Feature design. Since we label an instance between two different problem representations (i.e., BDD and MIP), the prediction should be a function of only problem-specific features. For the QSSP, we rely on the graph properties as well as attributes associated with the symmetric matrix Q. As the trained classifier is a component of the BDD-MIP algorithm and it is potentially invoked several times during search, we target features that can be efficiently computed for new instances that will be dynamically generated when exploring the solution space. For the graph properties, we select 10 features, namely, number of nodes (n), number of edges ($|\mathcal{E}|$), density (p), and seven features derived from node degrees, specifically, their mean, median, standard deviation (SD), maximum, minimum, the interquartile range (IR), and the variability score (SD/mean).

Conversely, for the features associated with Q, we select the percentage of positive coefficients in Q (v) and also characteristics associated with its main diagonal which is related to the linear profits w_i. For features coming from w_i, we define the mean, median, standard deviation (SD), minimum and maximum. Finally, we use a total of 16 features for the learning experiments.

Label definition. Since we cast the algorithmic question as a binary classification problem, we now define a procedure to binarize the label, which is a function on the performance of the two optimization technologies. Each QSSP instance is solved with both the MIP and a stand-alone BDD solver based on the representation proposed in this paper. For MIP, each QSSP example is solved with

CPLEX version `12.8` (5 times with a different random seed to deal with performance variability issues [31]) using its nonlinear programming-based branch and bound to get the MIP solving time MIP_{time} as the average of the five runs. Next, each instance is also solved with the stand-alone BDD solver (by setting $W = 128$ for all instances) to obtain the BDD_{time}. Finally, for each example, we assign the label `MIP` if the MIP solver is α times faster than the BDD solver (i.e., if $MIP_{time} \cdot \alpha \leq BDD_{time}$), otherwise we assign the label `BDD`.

Dataset composition. We assess the distribution of solving times and binarization parameter α to generate a dataset that is meaningful for the learning task. The dataset is composed of $12,500$ QSSP instances where $n \in \{20, 30, 40, 50, 60, 70, 80, 90, 100, 120\}$, $p \in \{10, 15, 20, 25, 30, 40, 45, 50, 60, 65, 70, 75\}$, $v \in \{25, 50, 75\}$, and label binarization parameter $\alpha = 5$. Finally, the number (resp., percentage) of instances labeled as `MIP` and `BDD` in the dataset is 9110 (72.88%) and 3390 (27.12%), respectively.

Supervised learning experiments. We construct the classifier using a Support Vector Machine (SVM) with RBF kernel [32], a classical supervised learning algorithm. We randomly split the dataset into training (75%) and test set (25%). To obtain features in the same range, we apply feature scaling and mean normalization so, each feature is normalized to have a mean of 0 and a standard deviation of 1. Each experiment consists of a training phase with 5-fold cross validation and grid search method for hyperparameter tuning, as well as a test phase on the neutral test set. We compare the SVM versus a dummy classifier (DUM), which follows a stratified strategy, i.e., it makes predictions based on the class distribution of the training set. The benchmark with DUM is a good practice as a measure of baseline performance for the trained classifier.

The learning methodology is implemented using Python with Scikit-learn [33]. Table 1 presents the standard performance measures of binary classification, namely, accuracy, precision, recall, and f1-score.

We highlight that the SVM classifier achieves high performance metrics. It compares favorably versus a dummy classifier corroborating that there is a

Table 1: Performance measures for the classifiers when predicting `MIP`/`BDD`

	DUM	SVM
Accuracy	0.607	0.976
Precision	0.257	0.953
Recall	0.289	0.954
F1-score	0.272	0.953

statistical pattern to be learned when discriminating either a MIP or BDD solver for tackling a QSSP instance. We remark that, for this particular application, false positives are computationally expensive, even more once a classifier is taken to production within the hybrid BDD-MIP. In this case, metrics different than accuracy could provide a better insight on the classifier performance. Precision, which is defined as the true positives divided by all positive predictions can be a good metric to analyze. A high precision indicates a low number of false positives, and we observe that the classifier presents a very good precision.

The classifier is able to capture enough about the discrimination from the selected features on training set to make meaningful predictions on test set. We conclude that the learning experiments support the selection and inclusion of the trained classifier within the BDD-MIP optimization algorithm.

4.2.3. BDD-MIP cutoff

Moreover, we incorporate another strategy used in [15]. We let LB denote a global incumbent solution obtained from the hybrid BDD-MIP exploration. Next, when each subproblem (associated with BDD node u) is explored through a MIP solver, i.e., the trained classifier predicts `MIP`, the corresponding BQP subproblem is modified by including the following constraint:

$$\sum_{i \in \mathcal{I}(u)} w_i x_i + \sum_{i \in \mathcal{I}(u)} \sum_{j \in \mathcal{I}(u) | i \neq j} 2 q_{ij} x_i x_j \geq LB - v^*(u). \tag{11}$$

Constraint (11) establishes a lower bound on the objective function for the subproblem, considering the global incumbent solution and the longest-path value from r to u. The lower-cutoff procedure already proved effective in [15] to speedup a hybrid BDD-MIP algorithm. In such a case, subproblems terminate significantly earlier the solving procedure, sometimes with the proof that no feasible solution meets the modified BQP conditions, also leading to prune the BDD node.

Algorithm 1 describes the hybrid BDD-MIP algorithm for the QSSP. At the beginning, the list of nodes to be explored consists of only the root node r. In line 2, while either L is not empty or the best bound is not less than or equal to the best incumbent solution we have to search for the optimal solution value. We take a BDD node u from L and evaluate in line 4, if the MIP-based pruning strategy should be applied to by calling, on-the-fly, the trained `MIP/BDD` SVM classifier described in Section 4.2.2. If the classifier predicts `MIP`, we use the MIP representation of the BDD subproblem and a MIP solver to prune the node in advance, updating the best incumbent if necessary. Otherwise, in line 9, we create a relaxed decision diagram rooted in u.

Next, if the BDD is exact we have found a feasible solution, update the incumbent solution if necessary and no further exploration is needed from the generated BDD. On the contrary (line 15), if the BDD is relaxed because the maximum width was reached when constructing any layer, we check if the node can be pruned by bound. If the best bound yielded by the relaxed BDD is greater than the incumbent solution, we must explore further by identifying an exact cutset C, and including the BDD nodes in C to L. The algorithm keeps exploring the solution space until the stopping criterion is met and the optimal solution is provided.

Algorithm 1 Hybrid BDD-MIP solver for the QSSP

```
    Input: QSSP instance
    Output: Optimal value
 1: Initialize list of nodes to be explored ($L$) with BDD root node
 2: while stopping criteria not met do
 3:     take node $u$ from $L$
 4:     if MIP-based pruning strategy applied to $u$ then
 5:         if best incumbent found then
 6:             update incumbent
 7:         Prune node $u$
 8:         continue
 9:     create relaxed BDD rooted in $u$
10:     if BDD is exact then
11:         if best incumbent found then
12:             update incumbent
13:         Prune node $u$
14:         continue
15:     if BDD is not exact then
16:         if Best bound greater than incumbent then
17:             Identify an exact cutset $C$
18:             for all nodes in $C$ do:
19:                 Add node to $L$
20:         create restricted BDD rooted in $u$
21:         if best incumbent found then
22:             update incumbent
23: return optimal value
```

Note that depending on the subproblem properties and hence the `MIP/BDD` classifier prediction, it may occur that no complementarity is identified by the algorithm, i.e., the MIP solver is never called to prune a BDD node (step 4 of

Algorithm 1). In such a case, the behavior of the hybrid algorithm actually reduces to a stand-alone decision diagram solving procedure.

5. Computational experiments on the QSSP

In this section, we benchmark the hybrid BDD-MIP algorithm against other general-purpose solvers coming from different optimization paradigms. In particular, we compare the hybrid approach versus two leading commercial MIP solvers with QP capabilities and a competitive SDP-based solver. We use `IBM-CPLEX 12.8` and `Gurobi 9.0.0` as the MIP solvers and we refer to them as CPLEX and Gurobi, respectively. We also implement the hybrid BDD-MIP approach in C++ and solve the MIP subproblems with CPLEX. All experiments are run on a Linux machine, Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz and 512 GB of RAM.

As it is described in [34], CPLEX can solve a BQP, such as the QSSP, in different ways. Let us denote the QSSP relaxation as the continuous problem where the integrality constraints (3), in Section 2, are relaxed. The semi-definiteness of matrix Q determines whether the QSSP relaxation is convex and therefore, the way CPLEX can tackle the problem.

In case the QSSP relaxation is convex, i.e., Q is positive semi-definite ($Q \succeq 0$), the problem can be solved by NLP-based branch and bound where a BQP relaxation is solved at each node of the search tree. Also in the convex case, CPLEX can linearize the BQP by transforming it into a MIP model by means of the McCormick inequalities and tackle the resulting formulation with standard MIP techniques. On the other hand, if the problem is not convex ($Q \not\succeq 0$), CPLEX has two alternatives. First, the problem can be convexified through the augmentation of the main diagonal of Q and then be solved by NLP-based branch and bound. Finally, CPLEX can also linearize the BQP and tackle the resulting larger MILP model via a traditional branch and bound. In the numerical experiments, we solve the QSSP using both alternatives i.e., linearizing and not linearizing the QSSP which can be simply selected through the CPLEX parameter `QToLin`[1]. The linearization mode allows us to evaluate the performance of a generic linearization technique (aiming at considering this approach as proposed in [12] where different linearization techniques are used to solve the QSSP). In addition, we tackle the instances using the CPLEX's NLP-based branch and bound algorithm whose performance has not been evaluated for the QSSP in the literature. For both modes, all internal CPLEX's capabilities (i.e., presolving, heuristics and cuts) and remaining default parameter settings are enabled.

We also solve the QSSP instances using Gurobi as MIP solver. The Gurobi version used in this experiments, like CPLEX, allows to solve to global optimality both convex and non-convex binary quadratic programmming models. As reported in its technical documentation, Gurobi implements a MIP solver where simplex and barrier algorithms tackle continuous BQPs. In addition, Gurobi's presolve may either convexify a problem by using bilinear constraints or linearize the problem so it can be solved by standard MIP techniques. Gurobi is also used with the default parameter settings and all internal capabilities (i.e., presolve, heuristics and cuts) enabled to make the comparison even more sound.

In addition to MIP solvers, BiqCrunch [11] becomes an interesting and natural alternative to be considered. This SDP-based branch and bound has also been used for the QSSP [35]. BiqCrunch is executed in the generic problem setup enabling internal heuristics 1, 2, and 3.

Regarding the Hybrid BDD-MIP solver, we use the MIP-based pruning strategy. The trained SVM classifier is invoked to label each BDD node subproblem as either `MIP` or `BDD` to define if the MIP-based pruning is performed. In addition, we implement the lower cutoff procedure and when the MIP solver is called to prune a BDD node, we solve the equivalent BQP subproblem using the clique formulation and CPLEX as MIP solver without any time limit. The variable ordering and exact cut selection strategies used are *the minimum number of states (MIN)* and *the last exact layer (LEL)*, respectively. We refer the reader to [13]

[1]Recently, CPLEX version `12.10` incorporates a ML algorithm to make this decision.

for details on different variable ordering heuristics and exact cutset strategies.

As mentioned before, the decision diagram width (W) is a critical parameter when manipulating limited-size DDs. Moreover, the mechanisms within the hybrid BDD-MIP approach generate an interesting dynamic between the DD-based branching scheme, the width, the bound quality, the size of the exact cutsets, the MIP-based pruning strategy, and the classifier predictions guiding the exploration. For instance, a very small width can dramatically deteriorate the bound quality, lead to a greater number of DD nodes to explore, and increase the computing time of the whole optimization process, up to one order of magnitude slower in the denser instances. Conversely, a very large width can improve the bounds and lead to a smaller number of DD nodes to explore. However, it can also trigger a greater computational effort when compiling each relaxed DD in the procedure which might not payoff when observing the whole computing time. Nevertheless, the effect of selecting a wrong value of W could be mitigated and indeed exploited in the hybrid BDD-MIP algorithm by calling more or less frequently the MIP-based pruning strategy. After exploring and exploiting this trade-off, the maximum width W for all instances is set to 64.

The testbed presented in these computational experiments corresponds to the dense instances used in the computational experiments in [12] plus, a set of sparser (and hence harder) instances which are also considered in [35]. The set of instances has number of nodes $n = \{100, 150\}$, density $p = \{25, 50, 75\}\%$, and percentage of positive coefficients in Q, $v = \{25, 50, 75\}\%$. The testbed considers 3 instances per combination (n, p, v) for a total of 54 instances. Every instance is processed five times by solving it with: BiqCrunch, CPLEX for the clique formulation (linearizing and not linearizing the BQP), Gurobi for the clique formulation, and finally, the proposed hybrid BDD-MIP solver. Each solver run uses only one thread with a time limit of $7,200$ seconds.

Table 2 compares the performance of the different solvers for the testbed instances. We present the average performance by combination (n, p, v) for a total of 18 different group of instances. Column 1 corresponds to the group id for the three instances of each combination (n, p, v). Columns 2, 3, and 4 present, for each group, the number of nodes, density, and percentage of positive coefficients in Q. Each solver is represented by a column where the base number corresponds to the average computational time employed for solving the 3 instances of the corresponding group. An exponent, in case it appears, indicates how many of those instances could not be solved to optimality within the time limit. In this way, column 5 is associated with BiqCrunch. For such an SDP-based solver, we generated the model from both MIP formulations, the edge and clique models. The edge formulation presents slightly better results and it is the one reported in these experiments. Columns 6 and 7 correspond to the performance of CPLEX solving the clique formulation of the problem both linearizing and not linearizing the QSSP, respectively. Next, column 8 corresponds to Gurobi's performance. For the MIP solvers, we use the clique formulation which reported 4% better results than the edge formulation. Column 9 relates to the performance of the hybrid BDD-MIP solver. Finally, column 10 shows the average speedup reached by the hybrid BDD-MIP solver with respect to the most competitive benchmark for each instance of the group. For example, the value in the seventh column "$6555.89^{(2)}$" of group #10 indicates that, within the time limit, CPLEX (when non linearizing the BQP) solved only 1 of the instances of the group $n = 150$, $p = 25$, $v = 25$, and the average solving time of the 3 instances is 6555.89 seconds. We present the detailed computational experiments for each instance in Appendix A.

We can observe from Table 2 that the hybrid solver outperforms all the benchmarks. The hardest instances are the sparsest ones ($p = 25$) where the speedups obtained with the hybrid solver are the smallest ones but greater than 1x. In addition, for group 12 ($n = 150 - p = 25 - v = 75$), the hybrid solver is the only one able to solve the 3 instances to optimality. On the other hand, when the instance is denser ($p = \{50, 75\}$) the BDD-based approach achieves remarkable speedups of at least one order of magnitude faster than the MIP solvers and the SDP-based solver.

Note that the percentage of positive coefficients v is related to the definiteness of matrix Q and evidently seems to be a very important feature of the

Table 2: Comparison between the hybrid BDD-MIP algorithm and other optimization paradigms for the QSSP

Instances				**BiqCrunch**	**CPLEX**		**Gurobi**	**Hybrid BDD-MIP**	**Speedup**
					Linearize	Non Linearize			
Group	n	p	v	CPU time (s)	CPU time (s)	CPU time (s)	CPU time (s)	CPU time (s)	vs. best benchmark
1	100	25	25	1139.31	272.65	113.13	84.75	24.50	3.50
2	100	25	50	7200.00$^{(3)}$	728.37	138.56	287.55	36.16	4.03
3	100	25	75	7200.00$^{(3)}$	1009.52	1388.85	184.00	167.02	1.10
4	100	50	25	829.49	18.28	9.67	11.09	0.72	13.53
5	100	50	50	4832.11	54.55	13.50	11.63	0.83	13.94
6	100	50	75	6677.62	89.59	30.51	14.37	0.74	19.50
7	100	75	25	461.94	3.89	4.18	1.03	0.06	18.34
8	100	75	50	1356.93	17.33	4.26	1.57	0.06	27.86
9	100	75	75	1162.46	30.49	6.12	1.71	0.05	32.28
10	150	25	25	7200.00$^{(3)}$	4776.18	6555.89$^{(2)}$	1607.86	560.80	2.87
11	150	25	50	7200.00$^{(3)}$	7200.00$^{(3)}$	7200.00$^{(3)}$	6918.76$^{(2)}$	888.37	7.92
12	150	25	75	7200.00$^{(3)}$	7200.00$^{(3)}$	7200.00$^{(3)}$	7200.00$^{(3)}$	2958.74	2.47
13	150	50	25	6901.33$^{(2)}$	130.06	135.64	46.01	7.45	6.19
14	150	50	50	7200.00$^{(3)}$	1184.23	182.24	96.15	7.39	13.03
15	150	50	75	7200.00$^{(3)}$	1752.94	582.37	91.46	6.83	13.38
16	150	75	25	2895.55	40.35	23.09	16.18	0.21	54.63
17	150	75	50	7160.97$^{(2)}$	79.11	26.49	43.05	0.19	139.37
18	150	75	75	7200.00$^{(3)}$	87.21	38.58	48.94	0.21	179.39
	Geom. Mean			3763.68	210.96	100.49	55.55	3.95	12.74

instance. Let us observe instances with $n = 150$ and $p = 50$ (i.e., groups 13, 14, and 15 in Table 2) to analyze this behavior. In such instances, if we observe the performance of the hybrid algorithm, no matter the value of v, the solving
550 time is almost equivalent for the three groups. However, this is not the case for CPLEX and Gurobi where the v value greatly impacts how the model is solved and hence the resulting performance. Noteworthy, the proposed BDD representation for the QSSP embeds the problem nonlinearity making no difference in the performance with respect to the semi-definiteness of matrix Q.

555 Figure 4 presents a performance profile to benchmark each solver in terms of the percentage of instances solved (out of the total 54 instances) within the execution time which ends at the time limit of $7,200$ seconds. Each of the five lines corresponds to one of the solvers considered in the comparison.

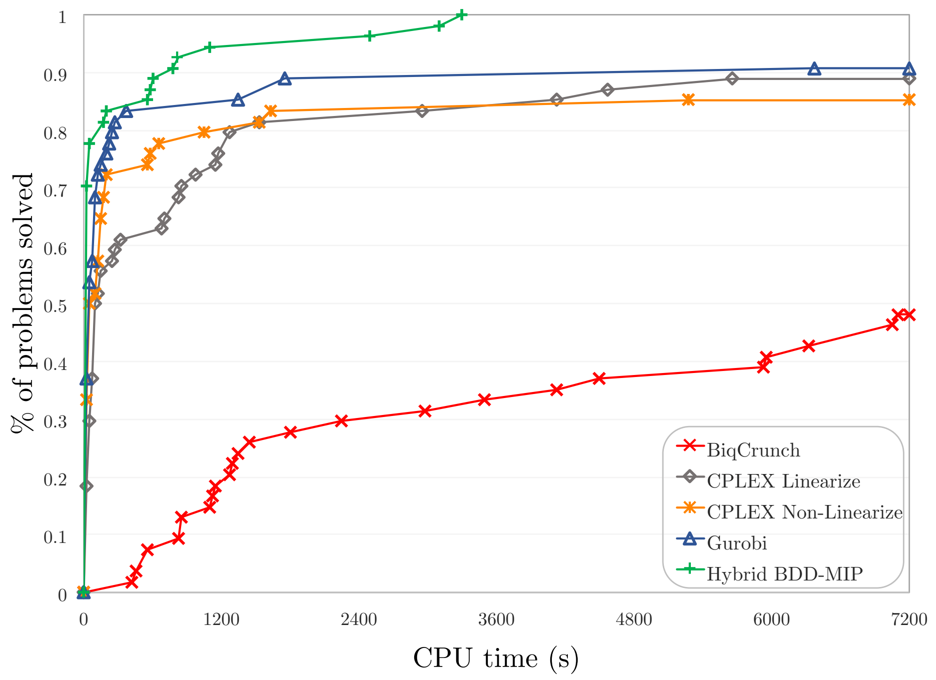


Figure 4: Performance profile for different solvers when tackling the QSSP

The performance profile shows the considerable dominance of the hybrid
560 BDD-MIP solver with respect to the other solvers. The hybrid approach is the

only method able to solve all instances long before the time limit is reached, solving each of the 54 instances within $3,300$ seconds.

Figure 5 compares the solution time of the hybrid BDD-MIP solver versus the solution time of the best benchmark across the different solvers (BiqCrunch, CPLEX linearizing, CPLEX non-linearizing, and Gurobi). We color instances by density value so that, green, red, and gray points correspond to $p = 25$, $p = 50$, and $p = 75$ instances, respectively. In a similar way, shapes are associated with the problem size n. Triangle and circle markers are related to $n = 100$ and $n = 150$ instances, respectively. A point located above the diagonal means that the hybrid BDD-MIP approach outperforms the best benchmark in such an instance for the corresponding $n - p$ group. As we use log scale to be able to represent all instances in one chart, some group of instances are represented by superimposed points, e.g., instances with $n = 150 - p = 50$ (red circles) appear represented by one red point to the most left side. Such a point indicates that the hybrid performs much better than the best benchmark.

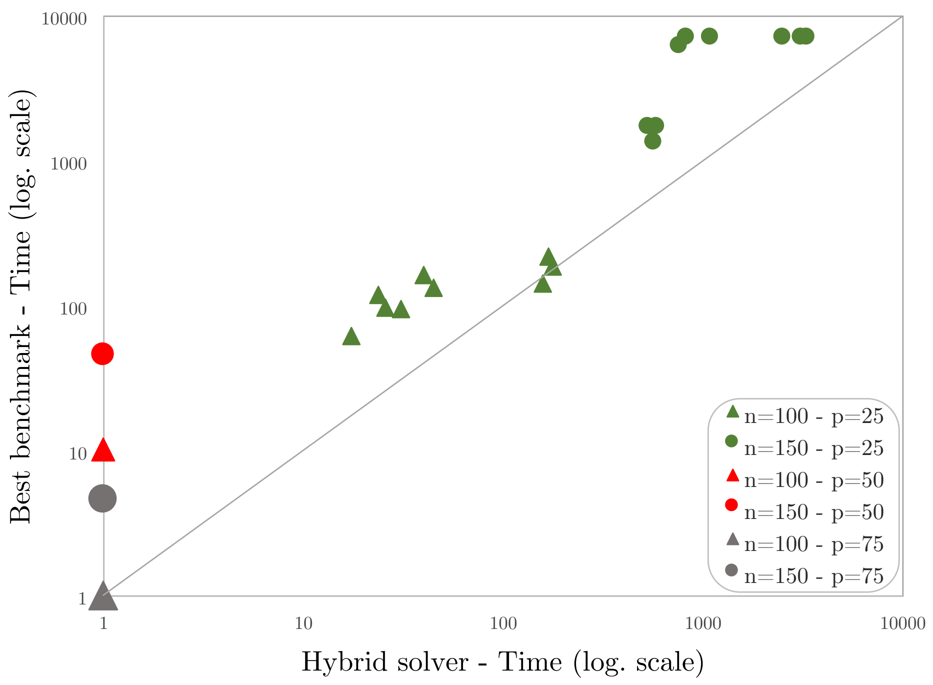
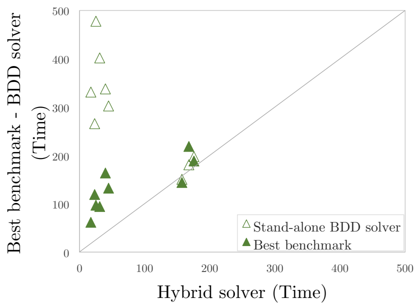


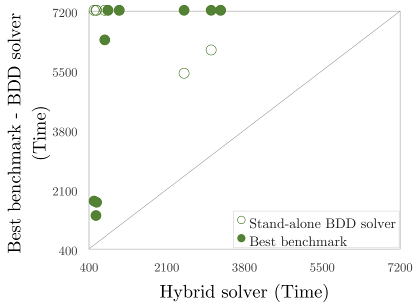
Figure 5: Hybrid BDD-MIP solver versus the best benchmark in terms of solution time per instance in log scale

Nevertheless, Figure 5 allows us to focus on the hardest instances, i.e., the sparsest ones with $p = 25$ (green markers). All but one instance are above the diagonal indicating that even for the hardest cases, the hybrid algorithm achieves a better performance than the benchmarks. For a fix density value, circle markers appear above triangle markers indicating in that case that larger instances are naturally harder.

Although, the BDD representation is one of the contributions in this paper, we evaluate the global effectiveness of the hybrid framework in comparison with a stand-alone BDD solver. We continue focused on the hardest (i.e., sparsest) subset of 18 instances with $p = 25$. Figure 6(a) and Figure 6(b) compare the solution time of the hybrid method versus a pure BDD solver but considering as well the best benchmark. As we analyze instances of the same density, solving times have the same magnitude and we generate the scatter plot not using the log scale. Figure 6(a) and Figure 6(b) are related to $n = 100$ and $n = 150$ instances, respectively. In both cases, instances related to the best benchmark are associated with filled markers, while those related to the stand-alone BDD solver correspond to markers with no fill.



(a) Instances $n = 100 - p = 25$



(b) Instances $n = 150 - p = 25$

Figure 6: Hybrid BDD-MIP solver versus both stand-alone BDD solver and the best benchmark for the sparsest instances

We can observe that the hybrid method consistently outperforms the stand-alone BDD solver. This is even more evident for large instances ($n = 150$) where the hybrid method really pays off. For some instances, we go from hitting the time limit with the pure BDD solver, to getting the best performance using the hybrid algorithm.

In a last computational experiment, we evaluate the performance of the BDD-based optimization approach on larger QSSP instances. We compare Gurobi, which is the best benchmark in the set of instances from the literature (Table 2), versus the hybrid BDD-MIP. The new set of instances has number of nodes $n = \{175, 200\}$, density $p = \{25, 50, 75\}\%$, and percentage of positive coefficients in Q, $v = \{25, 50, 75\}\%$. Similarly to the previous dataset, we consider 3 instances per combination (group) (n, p, v) for a total of 54 instances and 18 groups. Table 3 shows the computational experiments. The first four columns correspond to the instance information for each group. Column 5 and 6 report the performance of Gurobi and the hybrid BDD-MIP, respectively. Once again, if an exponent appears, it indicates how many of those instances were not solved to optimality within the time limit. Finally, column 7 presents the speedup by group. We present the detailed computational experiments for each instance in Appendix A.

For experiments reported in Table 3, the hybrid BDD-MIP uses the same trained classifier and maximum width W than in the previous experiment. Gurobi is also employed with the default parameter settings enabling all its internal capabilities to make the comparison more stringent. For $n = 175$, we note that Gurobi can solve only 1 out of 9 of the sparsest ($p = 25$) instances within time limit, whereas the Hybrid approach struggles with 2 (i.e., it solves 7 out of 9). When we increase the number of vertices to 200, we observe that neither Gurobi nor the Hybrid BDD-MIP can solve the challenging sparse instances. However, the lower and upper bounds obtained within time limit are, in general, better for the Hybrid BDD-MIP. Gurobi obtained slightly better lower bounds for only 3 out of the 81 instances, and this corresponds to the group $(200, 25, 25)$. Then, as the percentage of positive coefficients in Q increases, the bounds are more favorable for the Hybrid approach. For denser instances, no matter the size, the speedups show that BDD-based technology is significantly superior. It reaches, in some cases, more than 2 orders of magnitude faster computing times than a state-of-the-art commercial MIP solver. Instances of size 200 seem to be the current limit the Hybrid algorithm can reach within a 2-hour time limit on sparser instances. Instead, the method scales very well for instances of size 250 (and bigger) on denser graphs (see Table A.9 in the appendix).

Different elements of the hybrid BDD-MIP solver, such as the BDD-based branching scheme and, mainly, the black-box classifier guiding the exploration, make getting insights from the method's performance and evolution a complex task. However, we observe that the mechanisms of the hybrid BDD-MIP are mainly exploited for the sparsest (hardest) instances. On average, for instances with $p = 25$, the MIP solver prunes 91.56% of the total number of BDD nodes explored. In such a case, the MIP solver is frequently invoked indicating that the classifier possibly identifies a special pattern in sparse instances that leads

Table 3: Comparison between the hybrid BDD-MIP and Gurobi for QSSP instances with $n \in \{175, 200, 250\}$

Group	**Instances** n	p	v	**Gurobi** CPU time (s)	**Hybrid BDD-MIP** CPU time (s)	**Speedup**
19	175	25	25	$6821.25^{(2)}$	4132.60	1.70
20	175	25	50	$7200.00^{(3)}$	4686.14	1.55
21	175	25	75	$7200.00^{(3)}$	$6733.51^{(2)}$	1.08
22	175	50	25	117.35	14.97	7.89
23	175	50	50	223.18	24.01	9.31
24	175	50	75	260.92	19.90	13.14
25	175	75	25	58.07	0.35	166.56
26	175	75	50	75.74	0.35	216.22
27	175	75	75	83.60	0.37	222.73
28	200	25	25	$7200.00^{(3)}$	$7200.00^{(3)}$	1.00
29	200	25	50	$7200.00^{(3)}$	$7200.00^{(3)}$	1.00
30	200	25	75	$7200.00^{(3)}$	$7200.00^{(3)}$	1.00
31	200	50	25	263.15	37.91	6.92
32	200	50	50	576.81	54.58	10.39
33	200	50	75	504.95	44.11	11.65
34	200	75	25	107.89	0.53	203.68
35	200	75	50	128.78	0.53	245.02
36	200	75	75	161.21	0.56	290.24
	Geom. Mean			581.01	42.74	13.64

to the complementarity leveraged by the hybrid method. As the graph density increases, the MIP calls drastically decrease to the extent that the MIP-based pruning strategy is not employed in very dense instances. We can infer that calling the MIP solver for such instances could be a computationally expensive strategy. Nevertheless, the ML-based exploration seems to detect such a pattern.

6. Conclusions

We solved the quadratic stable set problem (QSSP) via BDD-based optimization contributing with both a BDD representation and an adapted hybrid BDD-MIP solver for the problem. We performed extensive computational experiments to compare the proposed hybrid method with other general optimization paradigms such as a semidefinite-based solver and two leading commercial MIP solvers with QP capabilities. We have shown that the hybrid BDD-MIP provides state-of-the-art results for solving the QSSP.

In addition, the BDD-based optimization technology shows high flexibility to represent quadratic problems. One important distinction of the proposed BDD representation is that it does not assume any special structure for the quadratic cost matrix which is an usual assumption in quadratic programming. Indeed, the proposed modeling to handle the quadratic profits in the state variables could be extended to any binary quadratic programming model where the Markov property holds in the conceptual DP model used to compile the decision diagram. We also contribute with a machine learning application to cast an algorithmic question within the hybrid BDD-MIP into a classification task. In an offline fashion, we train a classifier that is invoked, on-the-fly, by the hybrid algorithm to guide the exploration.

Therefore, when tackling an optimization problem, if both a representation from a different paradigm such as MIP or SDP and a mechanism to identify complementarity are available, a BDD-based hybrid approach could stand up as an effective solving method. This work opens up more research avenues where DD-based approaches, integrated methods, and machine learning are considered to tackle other quadratic combinatorial optimization problems.

7. acknowledgements

The authors thank the two anonymous referees for providing insightful suggestions to greatly improve this paper. The authors also express thanks to Fabio Furini and Emiliano Traversi for providing the instances from the literature used in Section 5.

References

[1] B. Balasundaram, S. Butenko, I. V. Hicks, Clique relaxations in social network analysis: The maximum k-plex problem, Operations Research 59 (1) (2011) 133–142. doi:10.1287/opre.1100.0851.

[2] S. Butenko, Maximum independent set and related problems, with applications, Ph.D. thesis, University of Florida, USA (2003).

[3] J. D. Eblen, C. A. Phillips, G. L. Rogers, M. A. Langston, The maximum clique enumeration problem: Algorithms, applications and implementations, in: J. Chen, J. Wang, A. Zelikovsky (Eds.), Bioinformatics Research and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 306–319.

[4] Q. Wu, J.-K. Hao, A review on algorithms for maximum clique problems, European Journal of Operational Research 242 (3) (2015) 693 – 709. doi:10.1016/j.ejor.2014.09.064.

[5] L. Xiaoli, W. Min, K. Chee-Keong, N. See-Kiong, Computational approaches for detecting protein complexes from protein interaction networks: a survey, BMC Genomics 11 (1) (2010). doi:10.1186/1471-2164-11-S1-S3.

[6] L. Cavique, A scalable algorithm for the market basket analysis, Journal of Retailing and Consumer Services 14 (6) (2007) 400 – 407. doi:10.1016/j.jretconser.2007.02.003.

[7] B. Jaumard, O. Marcotte, C. Meyer, Estimation of the quality of cellular networks using column generation techniques, GERAD Technical Report G-98-02 (1998).

[8] T. Hemazro, B. Jaumard, O. Marcotte, A column generation and branch-and-cut algorithm for the channel assignment problem, Computers & Operations Research 35 (4) (2008) 1204 – 1226. doi:10.1016/j.cor.2006.07.012.

[9] S. Karimi, P. Ronagh, A subgradient approach for constrained binary optimization via quantum adiabatic evolution, Quantum Information Processing 16 (8) (2017) 185. doi:10.1007/s11128-017-1639-2.

[10] F. Furini, E. Traversi, Hybrid SDP bounding procedure, in: V. Bonifaci, C. Demetrescu, A. Marchetti-Spaccamela (Eds.), Experimental Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 248–259.

[11] N. Krislock, J. Malick, F. Roupin, Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems, ACM Trans. Math. Softw. 43 (4) (2017) 32:1–32:23. doi:10.1145/3005345.

[12] F. Furini, E. Traversi, Theoretical and computational study of several linearisation techniques for binary quadratic problems, Annals of Operations Research 279 (1) (2019) 387–411. doi:10.1007/s10479-018-3118-2.

[13] D. Bergman, A. A. Cire, W.-J. v. Hoeve, J. Hooker, Decision Diagrams for Optimization, 1st Edition, Springer Publishing Company, Incorporated, 2016.

[14] D. Bergman, A. A. Cire, W.-J. van Hoeve, J. Hooker, Discrete optimization with decision diagrams, INFORMS Journal on Computing 28 (1) (2016) 47–66. doi:10.1287/ijoc.2015.0648.

[15] J. E. González, A. A. Cire, A. Lodi, L.-M. Rousseau, Integrated integer programming and decision diagram search tree with an application to the maximum independent set problem, Constraints 25 (1) (2020) 23–46. doi:10.1007/s10601-019-09306-w.

[16] A. A. Cire, W.-J. van Hoeve, Multivalued decision diagrams for sequencing problems, Operations Research 61 (6) (2013) 1411–1428.

[17] C. Tjandraatmadja, W.-J. van Hoeve, Target cuts from relaxed decision diagrams, INFORMS Journal on Computing 31 (2) (2019) 285–301. doi:10.1287/ijoc.2018.0830.

[18] T. Serra, A. U. Raghunathan, D. Bergman, J. Hooker, S. Kobori, Last-mile scheduling under uncertainty, in: L.-M. Rousseau, K. Stergiou (Eds.), Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Springer International Publishing, Cham, 2019, pp. 519–528.

[19] M. P. Castro, A. A. Cire, J. C. Beck, An MDD-based lagrangian approach to the multicommodity pickup-and-delivery TSP, INFORMS Journal on Computing (2019). doi:10.1287/ijoc.2018.0881.

[20] D. Bergman, A. A. Cire, Discrete nonlinear optimization by state-space decompositions, Management Science 64 (10) (2018) 4700–4720. doi:10.1287/mnsc.2017.2849.

[21] D. Bergman, L. Lozano, Decision diagram decomposition for quadratically constrained binary optimization, Preprint optimization-online (2018). URL http://www.optimization-online.org/DB_FILE/2018/10/6837.pdf

[22] M. Grötschel, L. Lovász, A. Schrijver, Stable sets in graphs, in: Geometric Algorithms and Combinatorial Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 1988, pp. 272–303. doi:10.1007/978-3-642-97881-4_10.

[23] S. Hosseinian, D. B. M. M. Fontes, S. Butenko, M. B. Nardelli, M. Fornari, S. Curtarolo, The maximum edge weight clique problem: Formulations and solution approaches, in: S. Butenko, P. M. Pardalos, V. Shylo (Eds.), Optimization Methods and Applications : In Honor of Ivan V. Sergienko's 80th Birthday, Springer International Publishing, Cham, 2017, pp. 217–237. doi:10.1007/978-3-319-68640-0_10.

[24] P. S. Segundo, S. Coniglio, F. Furini, I. Ljubić, A new branch-and-bound algorithm for the maximum edge-weighted clique problem, European Journal of Operational Research 278 (1) (2019) 76 – 90. doi:10.1016/j.ejor.2019.03.047.

[25] S. Hosseinian, D. B. M. M. Fontes, S. Butenko, A lagrangian bound on the clique number and an exact algorithm for the maximum edge weight clique problem, INFORMS Journal on Computing (2019). doi:10.1287/ijoc.2019.0898.

[26] S. Shimizu, K. Yamaguchi, S. Masuda, A branch-and-bound based exact algorithm for the maximum edge-weight clique problem, in: R. Lee (Ed.), Computational Science/Intelligence & Applied Informatics, Springer International Publishing, Cham, 2019, pp. 27–47. doi:10.1007/978-3-319-96806-3_3.

[27] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, USA, 1979.

[28] D. Bergman, A. A. Cire, Decomposition based on decision diagrams, in: C.-G. Quimper (Ed.), Integration of AI and OR Techniques in Constraint Programming, Springer International Publishing, Cham, 2016, pp. 45–54.

[29] Y. Bengio, A. Lodi, A. Prouvost, Machine learning for combinatorial optimization: a methodological tour d'horizon, Preprint arXiv:1811.06128 (2018).

[30] P. Erdös, A. Rényi, On the evolution of random graphs, in: Publications of the Mathematical Institute of the Hungarian Academy of Sciences, Vol. 5, 1960, pp. 17–61.

[31] A. Lodi, A. Tramontani, Performance variability in mixed-integer programming, in: Theory Driven by Influential Applications, INFORMS, 2014, pp. 1–12. arXiv:https://pubsonline.informs.org/doi/pdf/10.1287/educ.2013.0112, doi:10.1287/educ.2013.0112.

[32] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297. doi:10.1007/BF00994018.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830. URL http://dl.acm.org/citation.cfm?id=1953048.2078195

[34] P. Bonami, A. Lodi, G. Zarpellon, Learning a classification of mixed-integer quadratic programming problems, in: W.-J. van Hoeve (Ed.), Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Springer International Publishing, Cham, 2018, pp. 595–604.

[35] N. Krislock, J. Malick, F. Roupin, BiqCrunch numerical results (2019). URL https://www-lipn.univ-paris13.fr/BiqCrunch/results

Appendix A. Extended computational experiments associated with Table 2 and Table 3

Tables A.4, A.5, and A.6 present the extended computational experiments associated with the summarized Table 2 which is discussed in Section 5. Columns 1, 2, and 3 are associated to the instance properties, size n, density p, and percentage of positive coefficients in Q v, respectively. Columns 4 and 5 correspond to the instance id for a given $n-p-v$ and the optimal objective value. Columns 6, 7, 8, 9, and 10, present the solving time when using BiqCrunch, CPLEX linearizing, CPLEX non-linearizing, Gurobi, and the Hybrid BDD-MIP, respectively. The time limit is set to $7,200$ seconds. Finally, column 11 presents the speedup obtained by the BDD-MIP algorithm with respect to the best solving time between all the benchmarks.

Similarly, Tables A.7, A.8, and A.9 present the extended computational experiments on larger instances with size $n \in \{175, 200, 250\}$ associated with the summarized Table 3 which is discussed in Section 5. Columns 1, 2, and 3 are associated to the instance properties, size n, density p, and percentage of positive coefficients in Q (v), respectively. Column 4 corresponds to the instance id for a given $n-p-v$. Columns 5, 6, 7, and 8 are associated with Gurobi and present the best lower bound (LB), the best upper bound (UB), the optimality gap (calculated as (UB-LB)/UB), and the computing time. Columns 9, 10, 11, 12, 13, and 14 are related to the hybrid BDD-MIP. They present, in order from 9 to 14, the number of BDD nodes explored, the number of BDD nodes pruned by the MIP solver, the best lower bound (LB), the best upper bound (UB), the optimality gap, and the computing time, respectively. The time limit is set to $7,200$ seconds. Finally, column 15 presents the speedup obtained by the BDD-MIP algorithm with respect to Gurobi.

Table A.4: Comparison between the hybrid BDD-MIP algorithm and other optimization paradigms for the QSSP

Instance					**BiqCrunch**	**CPLEX**		**Gurobi**	**Hybrid BDD-MIP**	**Speedups**
						Linearize	Not Linearize			
n	p	v	#	Opt. Value	CPU (s)	CPU (s)	CPU (s)	CPU (s)	CPU (s)	vs. best benchmark
100	25	25	1	882	525.68	234.79	83.75	62.26	17.22	3.62
100	25	25	2	813	1106.77	318.86	130.86	97.60	25.49	3.83
100	25	25	3	792	1785.48	264.29	124.78	94.40	30.80	3.06
100	25	50	1	2576	7200.00	845.72	119.33	259.56	23.68	5.04
100	25	50	2	2468	7200.00	662.94	163.72	369.53	40.01	4.09
100	25	50	3	2509	7200.00	676.46	132.64	233.55	44.78	2.96
100	25	75	1	4758	7200.00	1257.45	1516.95	189.03	175.52	1.08
100	25	75	2	4877	7200.00	951.13	1606.76	218.82	168.34	1.30
100	25	75	3	5014	7200.00	819.99	1042.85	144.14	157.21	0.92
100	50	25	1	607	831.41	17.61	10.35	10.89	0.68	15.22
100	50	25	2	459	814.58	18.94	9.90	11.64	0.70	14.14
100	50	25	3	542	842.47	18.30	8.76	10.74	0.78	11.23
100	50	50	1	1163	4100.00	48.09	13.00	11.46	0.80	14.33
100	50	50	2	1138	4493.42	55.01	11.72	10.36	0.80	12.95
100	50	50	3	1002	5902.90	60.56	15.78	13.08	0.90	14.53
100	50	75	1	1829	7038.16	90.58	29.85	12.72	0.73	17.42
100	50	75	2	2047	5948.31	84.30	27.43	14.38	0.70	20.54
100	50	75	3	1748	7046.40	93.89	34.25	16.02	0.78	20.54

Table A.5: (Continued) Comparison between the hybrid BDD-MIP algorithm and other optimization paradigms for the QSSP

Instance					**BiqCrunch**	**CPLEX**		**Gurobi**	**Hybrid BDD-MIP**	**Speedups**
						Linearize	Not Linearize			
n	p	v	#	Opt. Value	CPU (s)	CPU (s)	CPU (s)	CPU (s)	CPU (s)	vs. best benchmark
100	75	25	1	267	527.32	5.04	4.68	1.01	0.05	20.20
100	75	25	2	378	440.87	3.19	3.86	0.97	0.06	16.17
100	75	25	3	273	417.62	3.45	4.00	1.12	0.06	18.67
100	75	50	1	483	1435.61	15.82	4.69	1.57	0.05	31.40
100	75	50	2	507	1335.20	18.28	3.96	1.52	0.06	25.33
100	75	50	3	525	1299.99	17.90	4.13	1.61	0.06	26.83
100	75	75	1	607	1255.21	29.69	5.95	1.66	0.05	33.20
100	75	75	2	757	1148.03	36.99	6.14	1.79	0.05	35.80
100	75	75	3	843	1084.15	24.79	6.28	1.67	0.06	27.83
150	25	25	1	1248	7200.00	5638.62	7200.00	1749.09	531.48	3.29
150	25	25	2	1229	7200.00	4117.49	5267.68	1726.29	585.53	2.95
150	25	25	3	1116	7200.00	4572.44	7200.00	1348.20	565.40	2.38
150	25	50	1	3322	7200.00	7200.00	7200.00	7200.00	824.74	8.73
150	25	50	2	3016	7200.00	7200.00	7200.00	7200.00	1078.56	6.68
150	25	50	3	3225	7200.00	7200.00	7200.00	6356.28	761.80	8.34
150	25	75	1	6962	7200.00	7200.00	7200.00	7200.00	2490.08	2.89
150	25	75	2	6438	7200.00	7200.00	7200.00	7200.00	3295.71	2.18
150	25	75	3	6332	7200.00	7200.00	7200.00	7200.00	3090.42	2.33

Table A.6: (Continued) Comparison between the hybrid BDD-MIP algorithm and other optimization paradigms for the QSSP

Instance					**BiqCrunch**	**CPLEX**		**Gurobi**	**Hybrid BDD-MIP**	**Speedups**
						Linearize	Not Linearize			
n	p	v	#	Opt. Value	CPU (s)	CPU (s)	CPU (s)	CPU (s)	CPU (s)	vs. best benchmark
150	50	25	1	665	6303.99	133.97	123.42	46.13	7.29	6.33
150	50	25	2	627	7200.00	134.50	135.07	44.77	7.85	5.70
150	50	25	3	642	7200.00	121.71	148.43	47.13	7.22	6.53
150	50	50	1	1500	7200.00	1166.65	168.72	82.57	7.03	11.75
150	50	50	2	1317	7200.00	1134.84	188.60	108.58	7.20	15.08
150	50	50	3	1322	7200.00	1251.19	189.40	97.30	7.94	12.25
150	50	75	1	2377	7200.00	804.81	540.03	89.54	6.52	13.73
150	50	75	2	2053	7200.00	2937.07	633.60	102.46	7.26	14.11
150	50	75	3	2293	7200.00	1516.94	573.47	82.39	6.70	12.30
150	75	25	1	451	2235.84	40.68	20.91	4.67	0.21	22.24
150	75	25	2	384	2971.97	34.10	23.21	4.61	0.21	21.95
150	75	25	3	347	3478.83	46.26	25.14	39.25	0.21	119.71
150	75	50	1	787	7200.00	79.54	25.75	44.67	0.18	143.06
150	75	50	2	873	7082.92	69.25	24.54	39.15	0.19	129.16
150	75	50	3	663	7200.00	88.54	29.18	45.32	0.20	145.90
150	75	75	1	895	7200.00	94.96	38.96	36.14	0.20	180.70
150	75	75	2	877	7200.00	99.28	37.58	65.96	0.22	170.82
150	75	75	3	888	7200.00	67.40	39.20	44.73	0.21	186.67

Table A.7: Comparison between the hybrid BDD-MIP and Gurobi for QSSP instances with $n \in \{175, 200, 250\}$

Instances				**Gurobi**				**Hybrid BDD-MIP**						**Speedups**
								BDD-nodes	subMIPs					
Group	n	p	v	Best LB	Best UB	Opt. Gap	CPU time (s)	explored	solved	Best LB	Best UB	Opt. Gap	CPU time (s)	
175	25	25	1	1298	1298	0	6063.74	100104	96669	1298	1298	0	2983.63	2.03
175	25	25	2	1118	2188	0.96	7200.00	136983	131702	1118	1118	0	4639.45	1.55
175	25	25	3	1043	2649	1.54	7200.00	138306	133879	1043	1043	0	4774.72	1.51
175	25	50	1	3207	9869	2.08	7200.00	117025	113546	3774	3774	0	4410.49	1.63
175	25	50	2	2803	9832	2.51	7200.00	109835	106377	2995	2995	0	5192.52	1.39
175	25	50	3	3671	9238	1.52	7200.00	114012	110094	3671	3671	0	4455.41	1.62
175	25	75	1	7680	15409	1.01	7200.00	83874	81458	7680	7680	0	5798.34	1.24
175	25	75	2	6349	16314	1.57	7200.00	45098	42883	6587	13787	1.09	7200.00	1.00
175	25	75	3	6482	16216	1.50	7200.00	53546	51560	6686	13022	0.95	7200.00	1.00
175	50	25	1	717	717	0	87.59	7771	0	717	717	0	15.29	5.73
175	50	25	2	895	895	0	119.58	5631	0	895	895	0	12.85	9.31
175	50	25	3	728	728	0	144.89	8112	0	728	728	0	16.77	8.64
175	50	50	1	1263	1263	0	227.98	15932	1	1263	1263	0	23.38	9.75
175	50	50	2	1351	1351	0	231.20	16065	6	1351	1351	0	23.79	9.72
175	50	50	3	1249	1249	0	210.36	17719	1	1249	1249	0	24.87	8.46
175	50	75	1	2298	2298	0	262.08	12384	9	2298	2298	0	19.13	13.70
175	50	75	2	2145	2145	0	251.34	14259	4	2145	2145	0	18.96	13.26
175	50	75	3	2335	2335	0	269.36	14161	3	2335	2335	0	21.62	12.46
175	75	25	1	371	371	0	53.68	255	0	371	371	0	0.34	157.88
175	75	25	2	402	402	0	61.77	246	0	402	402	0	0.33	187.18
175	75	25	3	409	409	0	58.75	260	0	409	409	0	0.38	154.61
175	75	50	1	642	642	0	88.12	373	0	642	642	0	0.36	244.78
175	75	50	2	621	621	0	62.12	308	0	621	621	0	0.35	177.49
175	75	50	3	713	713	0	76.97	285	0	713	713	0	0.34	226.38
175	75	75	1	900	900	0	102.81	528	0	900	900	0	0.40	257.03
175	75	75	2	917	917	0	72.70	387	0	917	917	0	0.35	207.71
175	75	75	3	1048	1048	0	75.28	376	0	1048	1048	0	0.37	203.46

Table A.8: (Continued) Comparison between the hybrid BDD-MIP and Gurobi for QSSP instances with $n \in \{175, 200, 250\}$

Instances				**Gurobi**				**Hybrid BDD-MIP**						**Speedups**
Group	n	p	v	Best LB	Best UB	Opt. Gap	CPU time (s)	BDD-nodes explored	subMIPs solved	Best LB	Best UB	Opt. Gap	CPU time (s)	
200	25	25	1	1233	5102	3.14	7200.00	98742	91409	1228	5495	3.47	7200.00	1.00
200	25	25	2	1229	5587	3.55	7200.00	91540	83594	1226	5651	3.61	7200.00	1.00
200	25	25	3	1255	5436	3.33	7200.00	89152	81142	1239	5612	3.53	7200.00	1.00
200	25	50	1	3171	16197	4.11	7200.00	72480	64032	3849	12251	2.18	7200.00	1.00
200	25	50	2	3136	16659	4.31	7200.00	63830	55759	3373	13135	2.89	7200.00	1.00
200	25	50	3	3001	15599	4.20	7200.00	73882	66552	3558	11944	2.36	7200.00	1.00
200	25	75	1	5850	25718	3.40	7200.00	20924	15712	7170	20751	1.89	7200.00	1.00
200	25	75	2	7556	24668	2.26	7200.00	33217	26997	7822	19175	1.45	7200.00	1.00
200	25	75	3	7448	25614	2.44	7200.00	35343	29168	7896	19121	1.42	7200.00	1.00
200	50	25	1	733	733	0	227.33	15466	0	733	733	0	36.39	6.25
200	50	25	2	755	755	0	252.20	14753	0	755	755	0	35.42	7.12
200	50	25	3	712	712	0	309.93	16279	0	712	712	0	41.92	7.39
200	50	50	1	1443	1443	0	813.54	30418	1	1443	1443	0	61.04	13.33
200	50	50	2	1433	1433	0	490.51	26524	0	1433	1433	0	52.33	9.37
200	50	50	3	1570	1570	0	426.39	25492	1	1570	1570	0	50.36	8.47
200	50	75	1	2550	2550	0	478.93	22468	9	2550	2550	0	46.65	10.27
200	50	75	2	2445	2445	0	522.66	19816	1	2445	2445	0	36.93	14.15
200	50	75	3	2410	2410	0	513.28	25190	8	2410	2410	0	48.76	10.53
200	75	25	1	484	484	0	106.55	351	0	484	484	0	0.52	204.90
200	75	25	2	390	390	0	100.98	394	0	390	390	0	0.54	187.00
200	75	25	3	546	546	0	116.14	251	0	546	546	0	0.53	219.13
200	75	50	1	905	905	0	129.93	374	0	905	905	0	0.54	240.61
200	75	50	2	956	956	0	112.08	367	0	956	956	0	0.53	211.47
200	75	50	3	714	714	0	144.32	399	0	714	714	0	0.51	282.98
200	75	75	1	1106	1106	0	158.00	558	0	1106	1106	0	0.61	259.02
200	75	75	2	1050	1050	0	153.22	539	0	1050	1050	0	0.56	273.61
200	75	75	3	1219	1219	0	172.42	339	0	1219	1219	0	0.51	338.08

Table A.9: (Continued) Comparison between the hybrid BDD-MIP and Gurobi for QSSP instances with $n \in \{175, 200, 250\}$

Instances				**Gurobi**				**Hybrid BDD-MIP**						**Speedups**
Group	n	p	v	Best LB	Best UB	Opt. Gap	CPU time (s)	BDD-nodes explored	subMIPs solved	Best LB	Best UB	Opt. Gap	CPU time (s)	
250	50	25	1	898	898	0	1149.65	34998	3	898	898	0	157.53	7.30
250	50	25	2	826	826	0	733.51	44231	1	826	826	0	192.68	3.81
250	50	25	3	874	874	0	1184.44	34912	1	874	874	0	164.12	7.22
250	50	50	1	1557	1557	0	2208.99	84926	3	1557	1557	0	428.82	5.15
250	50	50	2	1532	1532	0	2159.86	84933	4	1532	1532	0	463.70	4.66
250	50	50	3	1645	1645	0	2372.39	81951	8	1645	1645	0	432.17	5.49
250	50	75	1	2744	2744	0	2889.24	86741	77	2744	2744	0	476.84	6.06
250	50	75	2	2678	2678	0	2950.96	97142	77	2678	2678	0	427.36	6.91
250	50	75	3	2751	2751	0	2546.23	77539	30	2751	2751	0	416.00	6.12
250	75	25	1	560	560	0	447.82	429	0	560	560	0	1.27	352.61
250	75	25	2	446	446	0	356.40	723	0	446	446	0	1.42	250.99
250	75	25	3	426	426	0	369.74	775	0	426	426	0	1.50	246.49
250	75	50	1	771	771	0	422.91	1343	0	771	771	0	1.54	274.62
250	75	50	2	828	828	0	418.07	1135	0	828	828	0	1.41	296.50
250	75	50	3	709	709	0	670.93	2096	0	709	709	0	1.72	390.08
250	75	75	1	1103	1103	0	533.13	1543	0	1103	1103	0	1.52	350.74
250	75	75	2	1067	1067	0	501.15	2273	0	1067	1067	0	1.69	296.54
250	75	75	3	1176	1176	0	580.04	1367	0	1176	1176	0	1.40	414.31