Dynamic Node Packing

Christopher Muir, Alejandro Toriello
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia, USA 30332
chris dot muir at gatech dot edu, atoriello at isye dot gatech dot edu

November 7, 2020

Abstract

We propose a dynamic version of the classical node packing problem, also called the stable set or independent set problem. The problem is defined by a node set, a node weight vector, and an edge probability vector. For every pair of nodes, an edge is present or not according to an independent Bernoulli random variable defined by the corresponding entry in the probability vector. At each step, the decision maker selects an available node that maximizes the expected weight of the final packing, and then observes edges adjacent to this node. We formulate the problem as a Markov decision process and show that it is NP-Hard even on star graphs. Next, we introduce relaxations of the problem's achievable probabilities polytope, analogous to the linear and bilinear edge-based formulations in the deterministic case; we show that these relaxations can be weak, motivating a polyhedral study. We derive classes of valid inequalities arising from cliques, paths, and cycles. For cliques, we completely characterize the polytope and show that it is a submodular polyhedron. For both paths and cycles, we give an implicit representation of the polytope via a cut-generating linear program of polynomial size based on a compact dynamic programming formulation. Our computational results show that our inequalities can greatly reduce the upper bound and improve the linear relaxation's gap, particularly when the instance's expected density is high.

1 Introduction

The node packing problem, also known as the stable set or independent set problem, is a fundamental model in combinatorial optimization. Taking an undirected graph as input, the decision maker seeks the most valuable subset of pairwise non-adjacent nodes, either in terms of weight or cardinality. Node packing is well known to be equivalent to both the maximum clique and minimum vertex cover problems, and is also related to other classical problems, such as vertex coloring and set packing.

Node packing has received much attention from the optimization, operations research, graph theory, and computer science communities, in part due to its fundamental nature, but also owing

to its varied applications. One such important application area is scheduling [27, 50]. In the variant of single-machine job scheduling where each job has an associated time interval(s) in which it must be performed if accepted, the decision maker may construct a graph in which job-interval pairs are nodes, and nodes are adjacent when the corresponding job-interval pairs are conflicting, i.e. when they overlap in time or job. This idea is generalized by resource-constrained project scheduling models, where jobs require multiple resources and have more complex interactions, such as precedence requirements; an example of this type of model appears in [33].

Another important application area for node packing is wireless network design and operation. In large-scale wireless networks, it is often desirable to cluster nodes in order to facilitate more efficient communication. This is done by selecting *cluster heads* that drive the communication in and out of a cluster. Ideally, every node in a network should belong to a cluster and cluster heads should be out of each other's immediate range. The problem of optimally selecting cluster heads can be modeled as a weighted node packing problem [4]. Another example in wireless network design stems from selecting RFID readers to activate in a given network, as readers can interfere with each other if they both try to read from the same target. The problem of selecting which readers to activate at a given time to maximize the total number of read targets can be modeled as a weighted node packing problem [30].

Additional application examples include computer vision [10], train routing [52], coding theory [44], and military planning [29]. For a further discussion of applications, we refer the reader to [38, 48] and the references within.

In our proposed model, each node pair is associated with a probability that the corresponding edge appears in the graph. Nodes are selected dynamically, and after the decision maker commits to selecting a node, each potential edge between the chosen node and other nodes is sampled independently as a Bernoulli random variable with its corresponding probability. The decision maker's goal is to maximize the expected cardinality or weight of the resulting packing. This model generalizes classical deterministic node packing in two ways: It incorporates probabilistic graph topology akin to the Erdős-Rényi random graph model, and it introduces aspects of dynamic decision making.

One of our primary motivations for studying this model comes from dynamic scheduling problems arising in areas such as cloud computing. Consider a single-machine scheduling scenario in which jobs may have, for instance, known start times but random processing times that are only revealed once the job is scheduled. Therefore, we cannot construct an entire conflict graph beforehand, but we do have some probabilistic knowledge of possible edges present in the graph. This scheduling problem and others like it can be modeled using our dynamic node packing problem with additional modifications, such as allowing correlations between edges, and requiring the nodes to be considered in a particular order. As a first step, in this paper we consider a more fundamental model that does not have additional features.

Our contributions can be summarized as follows.

1. We propose a dynamic node packing model that incorporates uncertainty in the edge set;

the model is new, to our knowledge. We formulate the model as a Markov decision process and study its theoretical difficulty; in particular, it is NP-Hard even on star graphs.

2. We introduce basic relaxations of the model analogous to the standard linear and bilinear edge-based formulations of the deterministic node packing problem, and demonstrate that both relaxations can scale linearly with node count. Motivated by this negative result, we perform a polyhedral study of the dynamic node packing polytope and derive valid inequalities arising from cliques, paths and cycles.

3. We perform a computational study of the dynamic node packing problem, demonstrating the empirical effectiveness of our proposed inequalities. As a secondary contribution, our empirical results also reveal the remarkably good performance of a probabilistic and weighted generalization of the minimum-degree ordering heuristic.

The remainder of the paper is organized as follows. The next section includes a brief literature review. Section 3 introduces and formulates the problem, and gives preliminary results. Section 4 then includes our polyhedral study, with Section 5 detailing our computational study. Finally, Section 6 concludes.

2 Literature Review

The deterministic node packing problem has been studied extensively; in addition to being one of the most well known NP-Hard problems [26], it is also hard to approximate within a factor of $n^{1-\epsilon}$ [24]. Numerous algorithms have been proposed for solving the node packing problem: Current leading exact algorithms are primarily based on branch-and-bound, such as those compared in [32]. The methods generally differ based on how the upper- and lower-bounds are computed during the branching process. Heuristics are also often employed to solve the problem approximately; despite the negative approximability results, simple heuristics are observed to perform quite well in practice [49].

The node packing polytope has also been studied extensively as a way to derive polyhedral relaxations and bounds. Given a graph $G = (N, E)$ and associated weight vector $w \in \mathbb{R}^N_+$, the standard edge formulation for the node packing problem is

$$\begin{aligned} \max \ & \sum_{i \in N} w_i x_i & & \\ \text{s.t. } & x_i + x_j \leq 1 & & \forall \{i,j\} \in E \\ & x_i \in \{0,1\} & & \forall i \in N. \end{aligned}$$

The linear relaxation of this formulation is half-integral, meaning the extreme points of the corresponding polytope take on values in $\{0, 1/2, 1\}$. Furthermore, for the cardinality case, in any extreme point optimal solution of the relaxation, any variable taking value one also appears in at

least one optimal solution [36]. This edge formulation yields a weak linear relaxation in general, only capturing the integer hull in the case of bipartite graphs. Many polyhedral results for the node packing problem utilize specific types of graph structures to derive valid inequalities that strengthen the linear relaxation. Early results in this vein provided important classes of valid inequalities arising from cliques, odd holes, and odd antiholes [35, 37]. Additional examples include grilles [11], webs [47], and wheels [12].

The node packing problem may also be formulated as a bilinear programming problem,

$$\begin{aligned} \max \ & \sum_{i \in N} w_i x_i & & \\ \text{s.t. } & x_i x_j = 0 & & \forall \{i,j\} \in E \\ & 0 \leq x_i \leq 1 & & \forall i \in V. \end{aligned}$$

This model is commonly then relaxed to a semidefinite programming formulation. The value of this relaxation is equivalent to the Lováz ϑ function [31], which can then be used to solve the problem exactly in the case of perfect graphs [22]. Claw-free graphs are another special case where the problem is polynomially solvable [34]. For further information regarding the node packing polyhedron and node packing algorithms, we refer the reader to [41, 43, 51] and the references within.

Combinatorial optimization in the presence of uncertainty has received much attention in recent years, and many different ways of capturing uncertainty have been proposed. The models most closely related to ours are those in which some input parameters are only known probabilistically, and we are interested in static or adaptive algorithms that optimize the expected total reward. Examples of problems in which this type of paradigm has been studied include scheduling [40], equipment replacement [17], knapsack [6, 7, 8, 16], traveling salesman and vehicle routing [39, 45], and general packing [15]. These problems may be framed as Markov decision processes; for a general reference we refer the reader to [42].

Often in these models the curse of dimensionality makes exact methods intractable, motivating heuristics and bounding methods collectively known as approximate dynamic programming. The methods falling under this umbrella term vary significantly; the ones most relevant to our purposes include the study of polyhedra of achievable probabilities, e.g. [5, 13, 46], and approximate linear programming techniques, e.g. [1, 6, 14].

To the best of our knowledge, a dynamic model of this type has not been previously introduced for node packing. However, an alternative way of modeling uncertainty is via online models, where the decision maker optimizes against an adversarial input sequence. In particular, an online node packing model was proposed in [23]: Nodes are presented sequentially, and the decision maker must either irrevocably admit the node into the packing or permanently reject it. When a node is presented, it includes potential adjacencies to any previously admitted node. Traditional worst-case analysis shows that any deterministic algorithm in this setting has a competitive ratio of $1/(n-1)$. To address this, various relaxations and alternative models of the the online model

have been proposed, including multi-solution models [23], stochastic input models [21], models with advice [9, 19], and known graph models [18].

Another area of relevant research is node packing in random graphs. In the Erdős-Rényi random graph $G(n, 1/2)$, the largest node packing is almost surely in $\{2\log_2(n) - 1, 2\log_2(n)\}$, and a greedy algorithm almost surely finds a node packing of size $\log_2(n)$ [3]. Furthermore, the ϑ function is a loose bound on random graphs, almost surely giving a value of $\Theta(\sqrt{n})$ in $G(n, 1/2)$ [25]. In fact, the stronger Lovász-Schrijver hierarchy is also weak in random graphs; for $G(n, 1/2)$, after r iterations of the hierarchy, the value of the relaxation is almost surely $\Theta(\sqrt{n/2^r})$, and the hierarchy almost surely requires $\Theta(\log n)$ iterations for a tight bound [20].

3 Problem Statement and Preliminary Results

The *dynamic node packing* (DNP) problem is defined by a node set $N = \{1, \ldots, n\}$, probability vector $p \in [0,1]^{\binom{n}{2}}$, and node weight vector $w \in \mathbb{R}^N_+$. Each p_{ij} represents the probability that an edge is present between nodes $i, j \in N$; for convenience, we denote the complementary probability as $q_{ij} = 1 - p_{ij}$. When p_{ij} is constant across all pairs, we recover the Erdős-Rényi random graph model (we discuss this special case in detail below and in Section 4).

The decision maker constructs a packing sequentially, one node at a time. Upon selection of a node i, for every other remaining node $j \neq i$, the edge $\{i, j\}$ appears according to an independent Bernoulli random variable with probability p_{ij}. After the potential edges are sampled, the neighbor set of i, $\Gamma(i)$, is known exactly; the decision maker collects i's weight w_i and can no longer add any node $j \in \Gamma(i)$ to the packing. The process continues, with the decision maker selecting an available node until no eligible nodes remain. The objective is a policy that maximizes the expected total weight of the final packing. DNP may be formulated as an n-stage Markov decision process (MDP) defined on states $S \subseteq N$, with initial state N, and defined by the following recursion and boundary condition:

$$v^*_S = \max_{i \in S} \left\{ w_i + \mathbb{E}\left[v^*_{S \setminus (i \cup \Gamma(i))} \right] \right\}, \qquad \emptyset \neq S \subseteq N \tag{1a}$$

$$v^*_\emptyset = 0. \tag{1b}$$

The transition probabilities from state S to state S', given selection of node $i \in S$, are calculated as

$$\mathbb{P}(S'|S, i) = \begin{cases} \prod_{j \in S'} q_{ij} \prod_{j \in S \setminus (i \cup S')} p_{ij} & S' \subseteq S \setminus i \\ 0 & S' \not\subseteq S \setminus i. \end{cases}$$

In other words, the probability of transitioning from state S to S' after selecting node i is the probability of edges between i and nodes in $S \setminus (i \cup S')$ realizing, and of edges between i and nodes in S' not realizing.

The *value* linear program (LP) associated with the above recursion is then

$$\min v_N \tag{2a}$$
$$\text{s.t. } v_S - \mathbb{E}\left[v_{S\setminus(i\cup\Gamma(i))}\right] \geq w_i \qquad \forall \emptyset \neq S \subseteq N, \forall i \in S \tag{2b}$$
$$v_\emptyset \geq 0. \tag{2c}$$

The value LP provides a strong dual to the MDP. Specifically, the value v_N^* given by (1) is the optimal objective of (2), and any feasible solution provides an upper-bound on v_N^*.

The dual of (2) is known as the *policy* LP. Any feasible solution to the policy LP implies a policy in the MDP, with each variable corresponding to a state-action pair that can be interpreted as the probability of reaching that state and then selecting the corresponding action. The policy LP for the DNP is formulated as

$$\max \sum_{S\subseteq N}\sum_{i\in S} w_i x_{S,i} \tag{3a}$$
$$\text{s.t. } \sum_{i\in N} x_{N,i} = 1 \tag{3b}$$
$$\sum_{i\in S} x_{S,i} = \sum_{S'\supsetneq S}\sum_{j\in S'\setminus S} \mathbb{P}(S|S',j)x_{S',j} \qquad \forall \emptyset \neq S \subsetneq N, \forall i \in S \tag{3c}$$
$$x_{S,i} \geq 0 \qquad \forall S \subseteq N, \forall i \in S. \tag{3d}$$

As there are 2^n possible subsets of N, this LP contains $\Theta(n2^n)$ variables and $\Theta(2^n)$ constraints.

3.1 DNP on Star Graphs

In the general case, DNP is NP-Hard, as deterministic node packing is a special case in which all edge probabilities are binary. However, there are many cases in which the deterministic node packing problem is not only easy, but trivial. One such case is star graphs; a star graph is a tree of depth 1, a graph with one central root node (which we denote as 0 here) and n leaves connected only to the root. In the deterministic model, the decision maker may only select the root node if they do not select any leaf node and vice versa. The optimal objective value is then obviously $\max\{w_0, \sum_{i=1}^n w_i\}$, with the optimal solution being either to take the center node or to take all the leaf nodes.

Surprisingly, even on star graphs the DNP is NP-Hard; we prove this next. For simplicity of notation, in this section we use N to represent the leaf nodes only (and not the root), and denote the edges of the star graph by their corresponding leaf node, so that $\{0, i\}$ is simply i.

Observation 3.1. *Solving DNP on star graphs is equivalent to selecting the optimal subset $S \subseteq N$ of leaves to add to the packing before trying to add the root node 0.*

In other words, the DNP on star graphs reduces to a static subset selection problem. This observation stems from two facts: First, if the decision maker commits to taking a subset of leaf

nodes before the root, the order does not matter. Second, whenever the root node is added to the packing or is covered by one of its leaf nodes, any uncovered leaf may be added to the packing. This observation leads to the following formulation of the DNP on star graphs,

$$\max_{S\subseteq N}\left\{\sum_{i\in S} w_i + \left(1-\prod_{i\in S} q_i\right)\sum_{j\notin S} w_j + \prod_{i\in S} q_i\left(w_0 + \sum_{j\notin S} q_j w_j\right)\right\}. \tag{4}$$

The first term represents the immediate value of nodes added to the packing before trying to add the root node; the second term represents the probability-weighted value of the scenario in which at least one of the nodes in S covers the root node; and the third term represents the probability-weighted value of the root node being successfully added to the packing and then trying to add the leaf nodes in $N \setminus S$.

Theorem 3.2. *The DNP on star-graphs is NP-Hard.*

Proof. First, by rearranging and applying a natural logarithm, (4) is equivalent to

$$\max_{S\subseteq N}\left\{\sum_{i\in S} \ln q_i + \ln\left(w_0 - \sum_{k\in N} p_k w_k + \sum_{i\in S} p_i w_i\right)\right\}. \tag{5}$$

We show (5) is NP-Hard via a reduction from the partitioning problem. Given a set of numbers N with positive weights a_i, $i \in N$, the partitioning problem asks for a subset $S \subseteq N$ such that $\sum_{i\in S} a_i = \sum_{j\notin S} a_j$; without loss of generality, we assume that $\sum_{i\in N} a_i = 2$. By setting parameters for (5) as $q_i = e^{-a_i}$, $p_i = 1 - e^{-a_i}$, $w_i = a_i/(1-e^{-a_i})$, and $w_0 = \sum_{i\in N} a_i$, (5) becomes

$$\max_{S\subseteq N} -\sum_{i\in S} a_i + \ln(\sum_{i\in S} a_i).$$

Proceeding similarly to [2], a set $S \subseteq N$ with $\sum_{i\in S} a_i = 1$ exists if and only if the optimal solution to (5) is -1. □

This result is surprising in two ways. First, as mentioned before, node packing on star graphs is trivial in the deterministic case, but by simply adding uncertainty in the edge set the problem becomes NP-Hard. Second, the DNP on star graphs loses the dynamism of the general case; the order in which leaves are added to the packing only matters with respect to whether they are selected before attempting to add the root.

Our proof relies on a reduction from the partition problem, and thus only establishes that the DNP on star graphs is weakly NP-hard. We next sketch a pseudo-polynomial-time dynamic programming algorithm. In (5), we can interpret the inclusion of node i in S as paying an immediate cost of $-\ln q_i$ (since $\ln q_i$ is non-positive), and then receiving a terminal reward depending on the final value of $\sum_{i\in S} p_i w_i$. Assuming $w \in \mathbb{Z}^{N\cup 0}$ and that the probabilities are rational, let K be a scaling factor such that $K p_i w_i \in \mathbb{Z}$ for $i \in N$; note that K is bounded above by the least common multiple of the denominators defining the p_i's. We define states as (i, W) for $i = 1, \ldots, n+1$ and $W = 0, 1, \ldots, K\sum_{i\in N} p_i w_i$, where state (i, W) denotes the decision maker currently considering

node i with $K\sum_{j\in S} p_j w_j = W$. The initial state is $(1,0)$; letting $\sigma_{1,0}$ represent its optimal value, we get the recursion

$$\sigma_{i,W} = \max\{\ln q_i + \sigma_{i+1,W+Kp_iw_i}, \sigma_{i+1,W}\}$$
$$\sigma_{n+1,W} = \ln\left(w_0 - \sum_{k\in N} p_k w_k + W/K\right).$$

The recursion can be solved in $O(nK\sum_{i\in N} p_i w_i)$ time, and is therefore polynomial for instances where $K\sum_{i\in N} p_i w_i$ is polynomial in n; this holds, for example, if $K\max_i\{p_i w_i\}$ is a constant. Furthermore, because of the logarithm function, the recursion assumes exact real arithmetic; in practice this can be replaced with a numerical approximation of sufficient precision.

As a final note on the hardness result, the proof relies on varying both node weights and edge probabilities. If either the node weights or probabilities are uniform, the problem is polynomially solvable. If weights are uniform (the cardinality case), a trivial optimal solution is to take all leaf nodes before the root. If probabilities are uniform, suppose leaves are ordered by non-increasing value of weight, $w_1 \geq \cdots \geq w_n$. A simple exchange argument shows that an optimal set is of the form $\{1,\ldots,i\}$ for some $i \in N$; therefore, we only need to consider the $n+1$ sets of this form and select the one maximizing (4).

3.2 Relaxations of the DNP Polytope

The DNP can be solved using standard MDP techniques such as backwards induction or the value and policy LP's [42]. However, these exact approaches become intractable for even small instance sizes. Additionally, since DNP is NP-Hard for stars, it is unlikely efficient algorithms exist for any instance containing stars as induced subgraphs. One alternative is to consider relaxations of the DNP that may be computed efficiently and provide an upper bound on the true objective. Specifically, we are interested in relaxations of the polytope obtained by projecting the policy LP (3) into the space of *achievable* node probabilities, where each variable represents the probability of ever selecting the node. Denote this polytope by Q, and define it as

$$Q := \left\{z \in \mathbb{R}^N : \exists x \text{ satisfying (3b)–(3d) such that } z_i = \sum_{S\ni i} x_{S,i}\right\}.$$

That is, we consider relaxations of the problem

$$\max_{z\in Q} \sum_{i\in N} w_i z_i. \tag{6}$$

Note that Q is full-dimensional in $\mathbb{R}^N$. A simple relaxation of (6) is to consider a probabilistic version of the standard edge formulation of node packing. Given two nodes i, j, the probability of

selecting both of them is at most q_{ij}. Using this, we arrive at the relaxation

$$\max \sum_{i \in N} w_i z_i \tag{7a}$$

$$\text{s.t. } z_i + z_j \leq 1 + q_{ij} \qquad \forall i, j \in N \tag{7b}$$

$$0 \leq z_i \leq 1 \qquad \forall i \in N. \tag{7c}$$

Many of the inequalities included in this relaxation are necessary to describe Q. The non-negativity constraint in (7c) is facet-defining for Q for every $i \in N$, while the upper bound is facet-defining as long as $p_{ij} < 1$ for $j \in N \setminus i$. Similarly, for any pair of nodes i, j with $p_{ij} > 0$, constraints (7b) are facet-defining for Q as long as no $k \in N \setminus \{i, j\}$ has $p_{ik} = p_{jk} = 1$.

As in deterministic node packing, this relaxation is quite loose. For example, consider the cardinality DNP on the random graph $G(n, 1/2)$. A feasible solution for (7) sets $z_i = 3/4$, $i \in N$, which results in an objective value of $3n/4$. In other words, the objective scales linearly with n.

Unfortunately, unlike the deterministic case, passing to a bilinear relaxation is not much better. The relaxation analogous to the deterministic bilinear formulation is

$$\max \sum_{i \in N} w_i z_i$$

$$\text{s.t. } z_i z_j \leq q_{ij} \qquad \forall i, j \in N$$

$$0 \leq z_i \leq 1 \qquad \forall i \in N.$$

Using the same instance as before, a feasible solution sets $z_i = 1/\sqrt{2}$, $i \in N$, resulting in an objective value of $n/\sqrt{2} \approx 0.71n$, hardly improving on the linear relaxation; meanwhile, the largest node packing in $G(n, 1/2)$ is $\Theta(\log n)$ almost surely [3], and this quantity upper bounds the DNP since it allows the decision maker to observe the entire graph before choosing a packing. We discuss the DNP's optimal value for this instance in more detail in the next section.

The problem with both of these relaxations is that they only capture the probability relationships between pairs of nodes. It is possible to create tighter relaxations by introducing valid inequalities derived from larger structures within the instance; that is our goal.

4 Polyhedral Study

In this section we perform a polyhedral study of the polytope of achievable probabilities for the DNP. These inequalities can be added to the linear relaxation (7) to create a stronger upper bound. Specifically, we focus on two types of inequalities, those arising from cliques with uniform probabilities, and those arising from paths and cycles.

4.1 Cliques with Uniform Probabilities

A clique is a subset of pairwise adjacent nodes within a graph; we are interested here in the probabilistic analogue, node subsets for which every pair i, j has the same edge probability $p \in (0, 1)$, and call them a *probabilistic clique*. To differentiate such substructures that may appear in a larger instance from entire instances with this structure, we use K_n^p for the former, and keep the random graph notation $G(n, p)$ for the latter. We begin by showing that the DNP on $G(n, p)$ is efficiently solvable via a simple greedy policy.

Proposition 4.1. *The policy of selecting a highest-weight remaining node is optimal for DNP on $G(n, p)$.*

Proof. Take some optimal policy π and consider the first time it does not choose a highest-weight remaining node. Let S be the subset of remaining nodes, i be the node the policy selects, and j be a node with the highest remaining weight. We can write the expected value of the current state S as

$$w_i + \sum_{k \in S \setminus \{i\}} \mathbb{P}_\pi(k) w_k,$$

where $\mathbb{P}_\pi(k)$ is the probability of getting to pick node k under π. Now consider the policy π' that only differs from π in that we swap nodes j and i, selecting node j now and node i at any point where we would have selected j. The value of π' at subset S then becomes

$$w_j + \sum_{k \in S \setminus \{j\}} \mathbb{P}_{\pi'}(k) w_k.$$

By construction $\mathbb{P}_\pi(j) = \mathbb{P}_{\pi'}(i)$, the difference in value between the two policies at subset S is

$$w_j - w_i + \mathbb{P}_{\pi'}(i)(w_i - w_j) > 0.$$

So we can create a better policy by modifying π as described, a contradiction. □

Although straightforward, this result allows us to derive an efficient recursive formula for the expected value of the DNP on probability cliques. For the remainder of this section, we denote the expected value of the policy on $G(n, p)$ as $v_{n,p}$ in the cardinality case (where any available node is chosen, since $w_i = 1$ for all $i \in N$) , and $v_{n,p}^w$ in the weighted case. We also assume that nodes are labeled in non-increasing order of weight, so $w_1 \geq w_2 \geq \cdots \geq w_n$.

Proposition 4.2. *The expected value of the DNP on $G(n, p)$ in the cardinality case is defined recursively as*

$$v_{n,p} = 1 + \sum_{i=0}^{n-1} \binom{n-1}{i} p^i q^{n-1-i} v_{n-1-i,p},$$

where $q = 1 - p$, $v_{0,p} = 0$ and $v_{1,p} = 1$.

Proof. In the cardinality case, all nodes are equivalent up to relabeling. As such, it suffices to define the states of our recursion based on the number of nodes in the resulting subgraph, which at state n is a binomial random variable with parameters $n - 1$ and p. □

We can equivalently write $v_{n,p} = 1 + \mathbb{P}(2) + \cdots + \mathbb{P}(n)$, where again $\mathbb{P}(i)$ is the probability of adding i to the packing. Therefore, $v_{i,p} - v_{i-1,p} = \mathbb{P}(i)$; that is, the probability of adding i can be expressed as a difference in two expected values. With this we can compute the expected value of the weighted DNP using the following result.

Theorem 4.3. *For $G(n,p)$ and weight vector $w \in \mathbb{R}^N_+$ satisfying $w_1 \geq w_2 \geq \ldots \geq w_n$,*

$$v^w_{n,p} = w_1 + w_2(v_{2,p} - v_{1,p}) + \cdots + w_n(v_{n,p} - v_{n-1,p}).$$

Proof. It follows from Proposition 4.1 that we only need to consider the policy of selecting nodes in non-increasing weight order. We may then write the expected value as

$$v^w_{n,p} = w_1 + w_2\mathbb{P}(2) + \cdots + w_n\mathbb{P}(n).$$

Then by substituting $\mathbb{P}(i)$ with $v_{i,p} - v_{i-1,p}$ we arrive at our result. □

Theorem 4.3 and Proposition 4.2 allow for the direct computation of the optimal objective value of the DNP on probabilistic cliques. Proposition 4.2 also implies a class of valid inequalities for Q. For $G(n,p)$ and $S \subseteq N$, we obtain the inequality

$$\sum_{i \in S} z_i \leq v_{|S|,p}. \tag{9}$$

In addition to being valid for the clique itself, (9) is valid in the general case, when K^p_n is embedded in a larger instance, as the left side of the inequality is maximized when a policy attempts to add every node in the clique to the packing first. Having established their validity, a natural subsequent question concerns the facial dimension of the constraints.

Lemma 4.4. *For an instance given by $G(n,p)$, (9) is facet-defining for Q, for any $\emptyset \neq S \subseteq N$.*

Proof. Suppose first that $S = N$, and consider the matrix of points

$$\begin{bmatrix} 1 & (v_{2,p} - v_{1,p}) & (v_{3,p} - v_{2,p}) & \cdots & (v_{n,p} - v_{n-1,p}) \\ (v_{n,p} - v_{n-1,p}) & 1 & (v_{2,p} - v_{1,p}) & \cdots & (v_{n-1,p} - v_{n-2,p}) \\ (v_{n-1,p} - v_{n-2,p}) & (v_{n,p} - v_{n-1,p}) & 1 & \cdots & (v_{n-2,p} - v_{n-3,p}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (v_{2,p} - v_{1,p}) & (v_{3,p} - v_{2,p}) & (v_{4,p} - v_{3,p}) & \cdots & 1 \end{bmatrix}.$$

The rows correspond to the n cyclic shifts of the selection ordering $(1, 2, \ldots, n)$ and each satisfies (9) at equality. The above matrix is a circulant matrix, each entry is positive, and the sequence $(1, (v_{2,p} - v_{1,p}), (v_{3,p} - v_{2,p}), ..., (v_{n,p} - v_{n-1,p}))$ is monotonically decreasing. It follows from [28, Proposition 24] that the matrix is non-singular, i.e. we have n affinely independent points. As Q is full-dimensional, the result follows.

For $S \subsetneq N$, relabel nodes so that members of S appear before other nodes. We can construct an analogous $|S| \times |S|$ matrix by applying the policy only to nodes in S; we can then append zero entries for columns corresponding to $N \setminus S$ to get $|S|$ points satisfying (9) at equality. For the remaining $n - |S|$ rows, apply the policy corresponding to the ordering $(1, 2, \ldots, |S|)$ and then choose $i \in N \setminus S$ if it's still available. This creates a block-lower-triangular matrix; we have already argued above that the upper-left block is non-singular, and subsequent diagonal blocks are all of size one with positive entries equal to $(v_{|S|+1,p} - v_{|S|,p})$, thus the entire matrix is also non-singular, and the result follows. □

In addition to being valid in the general case when K_n^p appears as a substructure of a larger instance, the inequalities remain facet-defining under mild conditions.

Theorem 4.5. *Suppose a DNP instance contains K_n^p, and let S be its node set. Constraint (9) is facet-defining if and only if no node in $N \setminus S$ is adjacent to every node in S with probability 1.*

Proof. The proof is analogous to that of Lemma 4.4. For nodes $i \in N \setminus S$, we proceed as in the second part of the proof: Without loss of generality, suppose $p_{1i} < 1$. (Otherwise, permute nodes in S so the first one satisfies this property.) Then after applying the policy on S, i is still available with some positive probability, and thus the matrix is again non-singular. For the reverse direction, if some $i \in N \setminus S$ is connected to all nodes in S with probability 1, any point in Q that satisfies (9) at equality must have $z_i = 0$. □

Finally, we show that these inequalities are sufficient to describe the polytope of achievable probabilities.

Theorem 4.6. *For $G(n, p)$, Q is given by constraints (9) for $\emptyset \neq S \subseteq N$ and non-negativity constraints.*

Proof. We proceed by arguing that for any weight vector $w \in \mathbb{R}^N$, the optimal value is the consequence of an upper bound resulting from a conic combination of the mentioned constraints. First, we argue that we only need to consider weight vectors with positive entries. If there is an $w_i \leq 0$, in any optimal solution the we can set $z_i = 0$ and consider the problem in a lower dimension. So assume $w > 0$ and $w_1 \geq w_2 \geq \ldots \geq w_n$. Let z^* be optimal; the optimal expected value is

$$\begin{aligned}
\sum_{i=1}^{n} w_i z_i^* &= w_n \sum_{i=1}^{n} z_i^* + (w_{n-1} - w_n) \sum_{i=1}^{n-1} z_i^* + \ldots + (w_1 - w_2) z_1^* \\
&\leq w_n v_{n,p} + (w_{n-1} - w_n) v_{n-1,p} + \ldots + (w_1 - w_2) v_{1,p} \\
&= w_1 + (v_{2,p} - v_{1,p}) w_2 + \ldots + (v_{n,p} - v_{n-1,p}) w_n,
\end{aligned}$$

where the inequality follows from applying constraints (9) to each summand. Theorem 4.1 then implies that this inequality holds at equality for an optimal solution. □

Corollary 4.7. *For $G(n, p)$, Q is a submodular polyhedron. Constraints (9) can be separated in polynomial time via a greedy separation routine.*

Proof. The expected value $v_{n,p}$ is concave in n, and therefore $v_{|S|,p}$ is submodular in S. Given any $z \in [0,1]^N$, we can separate over constraints (9) by ordering z's coordinates in non-increasing order: Suppose we relabel nodes so that $z_1 \geq z_2 \geq \cdots \geq z_n$; then it suffices to check (9) for sets $\{1\}, \{1,2\}, \ldots, N$. □

In some instances, it may be unlikely to find a large clique with *uniform* probabilities, reducing the effectiveness of constraints (9). This can be partially remedied by relaxing the constraints: Given a clique with non-uniform probabilities, a valid constraint of form (9) can be generated using the minimum probability among any pair in the clique.

Naturally, there are other valid inequalities for the non-uniform case as well; the next section treats one such case.

4.1.1 Triangles

We next derive valid inequalities for triangles with possibly non-uniform edge probabilities. We denote the three nodes comprising the triangle as i, j, k and without loss of generality assume $p_{ij} \geq p_{jk} \geq p_{ik} > 0$. For triangles, any policy reduces to a selection ordering; furthermore, the policy's expected cardinality is in fact entirely determined by the first node selection, as we show next.

Proposition 4.8. *For triangles, a policy's expected cardinality is determined by the first node choice.*

Proof. Consider policies in which i is selected first. There are two orders in this case, i, j, k and i, k, j. The expected cardinalities of these selection orders are respectively

$$1 + q_{ij} + q_{ik}(p_{ij} + q_{ij}q_{jk}) = 1 + q_{ij} + p_{ij} - p_{ij}p_{ik} + q_{ij}q_{jk}q_{ik} = 2 - p_{ij}p_{ik} + q_{ij}q_{jk}q_{ik},$$

and

$$1 + q_{ik} + q_{ij}(p_{ik} + q_{ij}q_{jk}) = 1 + q_{ik} + p_{ik} - p_{ij}p_{ik} + q_{ij}q_{jk}q_{ik} = 2 - p_{ij}p_{ik} + q_{ij}q_{jk}q_{ik}.$$

That is, the expected size of the packing is the same for both orderings. □

This result implies that the following inequality is valid for any policy in which $z_i = 1$:

$$z_j + z_k \leq 1 - p_{ij}p_{ik} + q_{ij}q_{jk}q_{ik}.$$

We may combine this with $z_i = 1$ to rewrite the inequality as

$$z_i + (z_j + z_k)/\alpha_i \leq 2, \tag{10}$$

where $\alpha_i = 1 - p_{ij}p_{ik} + q_{ij}q_{ik}q_{jk}$. Inequality (10) is not necessarily valid for policies in which node i is not selected first. To create an inequality that is valid for the entire polytope, we lift the z_i

variable to create a new inequality of the following form

$$\beta_i z_i + (z_j + z_k)/\alpha_i \leq 1 + \beta_i. \tag{11}$$

We need a β_i in (11) satisfying

$$\beta_i \geq \frac{(z_j + z_k)/\alpha_i - 1}{1 - z_i} \tag{12}$$

for all feasible values of z_i, z_j, z_k.

Proposition 4.9. *Suppose $N = \{i, j, k\}$. Inequality (11) is valid for Q when*

$$\begin{aligned}\beta_i = \max\Bigg\{ & \frac{(1 + p_{jk})/\alpha_i - 1}{1 - q_{ij}(p_{jk} + q_{jk}q_{ik})}, \frac{(1 + p_{jk})/\alpha_i - 1}{1 - q_{ik}(p_{jk} + q_{jk}q_{ij})}, \\ & \frac{(1 + q_{jk}(p_{ik} + q_{ik}q_{ij}))/\alpha_i - 1}{(1 - q_{ik})}, \frac{(1 + q_{jk}(p_{ij} + q_{ij}q_{ik}))/\alpha_i - 1}{(1 - q_{ij})} \Bigg\}.\end{aligned} \tag{13}$$

Proof. We argue the inequality is valid for all orders. For orders in which i is selected first, the inequality is valid by construction. There are four possible orderings in which i is not selected first. Consider the two orderings in which node j is selected first; these result in the two points $(q_{ij}(p_{jk} + q_{jk}q_{ik}), 1, q_{jk})$ and $(q_{ij}, 1, q_{ik}(p_{ij} + q_{jk}q_{ik}))$. Similarly, the orderings in which k is selected first result in the points $(q_{ik}(p_{jk} + q_{ij}q_{jk}), q_{jk}, 1)$ and $(q_{ik}, q_{jk}(p_{ik} + q_{ij}q_{ik}), 1)$. Each of the four terms in (13) correspond to evaluating (12) at one of the four points. Therefore, taking the maximum over the four points results in a valid lifted inequality. □

A constraint of the form (11) can be generated for each node in the triangle, corresponding to that node being selected first. Note that if $p_{ij} = p_{jk} = p_{ik}$, this inequality is equivalent the corresponding clique inequality (9).

Proposition 4.10. *Constraints (11) are facet-defining for Q on triangles.*

Proof. It is easy to check that the inequality $z_i + (z_j + z_k)/\alpha_i \leq 2$ is facet-defining for the intersection of Q and the hyperplane defined by $z_i = 1$. The result then follows because we have applied maximal lifting to obtain β_i. □

Finally, we show how these constraints define Q on triangles, when added to the linear relaxation (7).

Theorem 4.11. *The constraints (11) generated for each first node choice i, j, k, along with (7b) and (7c), fully describe Q on triangles.*

Proof. We show how to construct the extreme point corresponding to the selection order i, j, k using the given constraints. The probability vector corresponding to this selection order is

$$(1, q_{ij}, q_{ki}(p_{ij} + q_{ij}q_{jk})).$$

Consider the constraints

$$z_i \le 1, \qquad z_i + z_j \le 1 + q_{ij}, \qquad \beta_i z_i + (z_j + z_k)/\alpha_i \le 1 + \beta_i.$$

Taking all three at equality, the first two imply $z_i = 1$ and $z_j = q_{ij}$. Substituting these into the third equation yields

$$\begin{aligned} z_k &= \alpha_i - q_{ij} = 1 - p_{ij}p_{ik} + q_{ij}q_{ik}q_{jk} - q_{ij} \\ &= p_{ij}p_{ik} - p_{ij} + q_{ij}q_{ik}q_{jk} = p_{ij}q_{ik} + q_{ij}q_{ik}q_{jk} = q_{ik}(p_{ij} + q_{ij}q_{jk}). \end{aligned}$$

The argument follows similarly for the other orderings.

For the extreme points in which zero, one, or two nodes are selected with non-zero probability, e.g. $(0,0,0)$, $(1,0,0)$, $(1,q_{ij},0)$, it is simple to verify that they are already extreme points of the polytope defined by only (7b) and (7c). □

4.2 Paths

In our context, a (probabilistic) path is a non-repeating sequence of nodes where each node is adjacent to the node that precedes it with positive probability. In this section we analyze the DNP on paths and use it to derive a cut-generating LP based on paths for the general case. We also extend the approach to cycles below.

Given an n-node path with nodes labeled in the path's order, $1, 2, \ldots, n$, we denote the optimal expected value of the DNP on the path as $u_{1,n}$. Similarly, for any i-j sub-path, $i \le j$, we use $u_{i,j}$. Because there are only quadratically many sub-paths, the recursion (1) simplifies to

$$u_{i,j} = \max_{i \le k \le j} \{w_k + p_{k-1,k}u_{i,k-2} + q_{k-1,k}u_{i,k-1} + p_{k,k+1}u_{k+2,j} + q_{k,k+1}u_{k+1,j}\} \qquad 1 \le i \le j \le n \qquad (14a)$$

$$u_{i,j} = 0 \qquad i > j. \qquad (14b)$$

In the recursion, we consider which node in a path to choose; this necessarily cuts the path in two, but it may do so in several ways, depending on whether the node's edges realize or not.

Recursion (14) runs in $O(n^3)$ time, as there are $O(n^2)$ sub-paths and we consider $O(n)$ node choices for each. As a consequence, we can efficiently solve the LP formulation

$$\begin{aligned} &\min u_{1,n} \\ &\text{s.t. } u_{i,j} - p_{k-1,k}u_{i,k-2} - q_{k-1,k}u_{i,k-1} - p_{k,k+1}u_{k+2,j} - q_{k,k+1}u_{k+1,j} \ge w_k \qquad 1 \le i \le k \le j \le n \\ &\qquad u_{i,j} = 0 \qquad i > j, \end{aligned}$$

and its dual,

$$\max \sum_{1 \le i \le k \le j \le n} w_k x^k_{i,j} \qquad (15a)$$

$$\text{s.t.} \sum_{k=1}^{n} x_{1,n}^k = 1 \tag{15b}$$

$$\sum_{k=i}^{j} x_{i,j}^k = \sum_{\ell=1}^{i-1} q_{i-1,i} x_{\ell,j}^{i-1} + \sum_{\ell=1}^{i-2} p_{i-2,i-1} x_{\ell,j}^{i-2} + \sum_{\ell=j+1}^{n} q_{j,j+1} x_{i,\ell}^{j+1} + \sum_{\ell=j+2}^{n} p_{j+1,j+2} x_{i,\ell}^{j+2} \qquad 1 \le i \le j \le n \tag{15c}$$

$$x_{i,j}^k \ge 0 \qquad \forall 1 \le i \le k \le j \le n. \tag{15d}$$

In (15), $x_{i,j}^k$ represents the probability of encountering the i-j path and choosing node k.

As with cliques, our main goal is not to solve the DNP on paths, but rather to derive valid inequalities we can apply to the general problem. We summarize this in the next result.

Theorem 4.12. *Let $\hat{z} \in [0,1]^N$. Then $\hat{z} \in Q$ for the 1-n path if and only if the following LP has optimal value equal to zero:*

$$\min u_{1,n} - \sum_{i=1}^{n} \lambda_i \hat{z}_i \tag{16a}$$

$$\text{s.t. } u_{i,j} - p_{k-1,k} u_{i,k-2} - q_{k-1,k} u_{i,k-1} - p_{k,k+1} u_{k+2,j} - q_{k,k+1} u_{k+1,j} \ge \lambda_k \quad \forall 1 \le i \le k \le j \le n \tag{16b}$$

$$u_{i,j} = 0 \qquad \forall i > j. \tag{16c}$$

If the LP is unbounded, any ray $(\hat{\lambda}, \hat{u})$ with negative objective value generates the valid inequality

$$\sum_{i=1}^{n} \hat{\lambda}_i z_i \le \hat{u}_{1,n},$$

which cuts off $\hat{z}$.

Proof. As in the general case (6), $\hat{z}$ is an achievable probability if some $\hat{x}$ feasible in (15) satisfies

$$\hat{z}_k = \sum_{\substack{i \le k \\ j \ge k}} \hat{x}_{i,j}^k, \qquad k \in N.$$

Applying Farkas' lemma to this equation and the feasible region of (15) yields (16). □

In the general DNP, for any fixed path we may iteratively improve the linear relaxation using (16) to generate cutting planes. Next, we discuss extending this approach to cycles.

4.2.1 Cycles

The probabilistic path concept naturally extends to cycles, and our approach can also capture Q for cycles in a cut-generating LP. Extending our path notation, we now suppose we have a probabilistic cycle with nodes $1, \ldots, n$, where i is connected to $i+1$ with positive probability and 1 is connected to n with positive probability as well. To hopefully ease the notational burden in

our exposition, we adopt in this section the convention that all indices use modular arithmetic; for example, 0 is identified with n and $n+1$ is identified with 1. Note also that we address the special case of triangles above; we are able to describe Q for triangles explicitly, while here we give an implicit description that applies to any cycle. Therefore, we assume without loss of generality that $n \geq 4$.

The recursion (14) can be extended to cycles. After the decision maker selects a first node, the resulting subgraph is in fact a path, and therefore after the first decision the recursion proceeds exactly as in that case. The only difference is that this path may have a starting node with a higher index than its end node, so we adopt modular arithmetic in our notation. Define the optimal expected value of the cycle as u_N, and extend our previous notation to $u_{i,j}$ for any $i, j \in N$. In particular, if $i \leq j$, this corresponds to the path $(i, i+1, \ldots, j)$, whereas if $i > j$, the variable corresponds to the path $(i, i+1, \ldots, n, 1, 2, \ldots, j)$. We then obtain the recursion

$$
\begin{aligned}
u_N &= \max_{i \in N} \{ w_i + p_{i-1,i} p_{i,i+1} u_{i+2,i-2} + q_{i-1,i} p_{i,i+1} u_{i+2,i-1} + p_{i-1,i} q_{i,i+1} u_{i+1,i-2} + q_{i-1,i} q_{i,i+1} u_{i+1,i-1} \} \\
u_{i,j} &= \max_{i \leq k \leq j} \{ w_k + \mathbb{1}_{k \geq i+2} p_{k-1,k} u_{i,k-2} + \mathbb{1}_{k \geq i+1} q_{k-1,k} u_{i,k-1} \\
&\qquad + \mathbb{1}_{k \leq j-2} p_{k,k+1} u_{k+2,j} + \mathbb{1}_{k \leq j-1} q_{k,k+1} u_{k+1,j} \} \qquad , \qquad i \neq j \\
u_{i,i} &= w_i, \quad i \in N.
\end{aligned}
$$

We use indicator functions to avoid notational overlap, but otherwise the recursion is similar to (14), with the exception of u_N. The complexity remains $O(n^3)$, as we have only doubled the number of paths. From this recursion, we can proceed exactly as in the path case to derive a cut-generating LP to use in the general case, when cycles appear as substructures. Moreover, although this LP doubles the number of variables, we significantly increase the number of cutting planes we can generate, since a single cycle of length n captures n different paths with n nodes.

5 Computational Study

The primary objective of our computational experiments is to evaluate the effectiveness of inequalities derived in our polyhedral study, in terms of their bound improvement over the basic linear relaxation (7). To benchmark these various bounds, we also study a probabilistic extension of the *minimum degree ordering heuristic*; see, e.g. [49]. At any state $S \subseteq N$, the heuristic chooses

$$
\operatorname*{argmax}_{i \in S} \left\{ w_i - \sum_{j \in S \setminus i} p_{ij} w_j \right\}.
$$

In other words, the heuristic adds a node to the packing that maximizes the immediate net expected value, where the positive element of the value is the chosen node's weight, while the negative element is the probabilistically discounted value of the node's possible neighbors. In the cardinality case, this reduces to choosing the node with the minimum expected degree in S, and

further reduces precisely to the minimum degree ordering in the deterministic case. We refer to this heuristic as the *max expected weight heuristic* (MEWH) in our experiments.

For instances in which all non-zero probabilities are uniform, we also define a dual-based value function approximation heuristic. Specifically, the heuristic approximates the expectation in the value function (1) by the dual prices of probabilistic clique constraints at an optimal solution of (7) strengthened with constraints (9). Let $\mathcal{C}$ be the set of probability cliques and y be the corresponding vector of dual prices; at any state $S \subseteq N$ we select a node according to

$$\operatorname*{argmax}_{i \in S} \left\{ w_i + \sum_{C \in \mathcal{C}} y_C \mathbb{E}\left[v_{|C \cap (S \setminus (i \cup \Gamma(i)))|, p} \right] \right\},$$

where p again denotes the uniform edge probability and $v_{n,p}$ is defined in Proposition 4.2. Each expectation is calculated as

$$\mathbb{E}\left[v_{|C \cap (S \setminus (i \cup \Gamma(i)))|, p} \right] = \begin{cases} v_{|C \cap S|, p} - 1 & i \in C \\ \displaystyle\sum_{j=0}^{|\gamma(i) \cap C \cap S|} \binom{|\gamma(i) \cap C \cap S|}{j} p^j (1-p)^{|\gamma(i) \cap C \cap S| - j} v_{|C \cap S| - j, p} & i \notin C, \end{cases}$$

where $\gamma(i) \subseteq N \setminus i$ denotes nodes adjacent to i with probability p. Recall that although $\mathcal{C}$ may contain an exponential number of probabilistic cliques, an optimal extreme point solution of the LP has at most n corresponding constraints with non-zero dual prices. This method can be viewed as maximizing the immediate reward of the chosen node's weight and the expectation of the sum of dual prices multiplied by the expected size of a packing in the appropriate probability clique. The multipliers $v_{|C \cap S|, p}$ scale dynamically as nodes are packed and covered. We refer to this heuristic as the *dual clique heuristic* (DCH). In addition to providing another benchmark, the DCH allows us to test our tightened relaxation's potential to guide heuristic policies.

As a final benchmark, we also consider the expected value of an instance's max-weight node packing, the generalization of the expected max-cardinality packing in random graphs. Unlike the DNP, this benchmark allows the decision maker to observe the entire graph's realization before selecting a packing. In stochastic programming terms, we allow the decision maker to violate non-anticipativity by giving them earlier access to information. Equivalently, it is the "hindsight-optimal" value, what the decision maker would have liked to do with the benefit of hindsight; therefore, we refer to it as HSO below. Whereas our polyhedral bounds are computed via LP, the MEWH, DCH, and HSO must be approximated via simulation.

5.1 Instances

We conduct experiments using two types of instances designed to evaluate the performance of our bounds and benchmarks on sparser and denser instances. For dense instances, we use random graphs to generate a topology. Specifically, to generate each instance, we sample a random graph from $G(100, p_1)$, and assign each realized edge the uniform probability p_2, where

$p_1, p_2 \in \{0.5, 0.75, 0.9\}$. For each of the nine value pairs of p_1 and p_2, we generate ten instances in this manner, all with the cardinality objective, yielding 90 total instances, each with 100 nodes. Note that the effect of the two probability parameters is multiplicative: In an instance's *realization*, the probability that any node pair will have an edge is $p_1 p_2$, which ranges between 0.25 and 0.81.

For sparse instances, we use binary trees. Each instance's topology is given by a full binary tree of depth six, which has $2^6 = 64$ leaves and $2^7 - 1 = 127$ total nodes. We assign node weights uniformly at random from $[1, 10]$, and each edge in the tree has uniform edge probability $p \in \{0.5, 0.75, 0.9\}$. For each value of p, we generate ten instances that only vary in their node weights, for a total of 30 instances.

5.2 Experiments

For dense instances, we test four bounds: Relaxation (7), (7) strengthened by path and cycle cuts, relaxation (7) strengthened by probabilistic clique cuts, and (7) strengthened by path, cycle, and clique cuts.

For each instance, we generate path and cycle cuts by greedily searching for 200 paths of maximum length ten and then, if possible, connecting the start and end of the path to form a cycle. At each iteration, given a current probability vector, we solve (16) for each path and the analogous LP for each cycle, and then re-optimize. We use three stopping criteria: Either we find no new cuts, the absolute change in objective is less than 0.5, or we reach a maximum of 50 iterations. We elected to generate path and cycle cuts in this manner after exploratory tests suggested better bounds came from many shorter paths, rather than a smaller number of longer paths.

To compare probabilistic cliques, for each topology we compute the 5,000 largest maximal probabilistic cliques, resulting in 5,000 initial cuts. We then re-optimize, adding violated cuts (9) for node subsets until we find none, or reach the same alternative termination conditions as with paths and cycles. For the experiments that combined the two inequality classes, we perform both types of cut generation in each each iteration.

For the sparse instances, since their topology is based on binary trees, we can only compare the relaxation (7) and (7) with path inequalities. However, unlike in the dense instances, we can enumerate all maximal paths (corresponding to all pairs of leaves), and thus add any violated path inequality using (16); this is what we do in the experiments until we find no more violated inequalities or, again, the alternate termination criteria are met.

For the simulation-based benchmarks, we calculate each by simulating 100 realizations of the instance and taking sample averages. For MEWH, this simply amounts to running the heuristic on each realization. For DCH, we use the optimal solution to the relaxation (7) with probabilistic clique constraints added, and also run the heuristic on each realization. We only test the DCH on dense instances, as the sparse instances have no probabilistic cliques beyond node pairs. For HSO, we solve a deterministic node packing problem on each realization; for dense instances, this entails solving 100 node packing integer programs (IP), while for the sparse instances the realization is guaranteed to be a forest, and thus an LP suffices.

We conduct all experiments on a MacBook with a 2.7GHz Dual-Core Intel i5 processor. Our base code and simulation use Python 3.7.3, and the LP and IP solves use Gurobi 9.0. For all instances, the solution of (7) takes less than one second. For dense instances, adding path and cycle inequalities to the bound increases the solve time to between 10 and 20 seconds, while adding probability clique inequalities raises solution times to between five seconds and five minutes, generally increasing with the size of the cliques. For the simulation-based benchmarks, the entire MEWH simulation runs in one or two seconds, the DCH takes approximately one minute per instance, while the HSO simulation takes an average of three minutes per instance. On the sparse instances, adding path inequalities to (7) results in solve times of about three minutes, as the number of paths to check is $\binom{64}{2} = 2,016$.

5.3 Summary of Results

Table 1 below presents average results for the dense instances. We report results as the geometric mean of the ratio between the corresponding value and the HSO bound over the 10 instances of each type; the Appendix includes detailed results. We choose this presentation because the HSO bound is independent of our proposed methods and such bounds are known to be strong for many dynamic problems.

Table 1: Geometric mean and standard deviation of dense instance results as ratios of the HSO bound.

p_1	p_2	(7)	(7) + P/C	(7) + Clq.	(7) + P/C + Clq.	MEWH	DCH
0.5	0.5	4.45 ± 0.05	3.93 ± 0.04	2.29 ± 0.01	2.29 ± 0.01	0.83 ± 0.01	0.74 ± 0.01
0.5	0.75	5.18 ± 0.04	4.63 ± 0.04	2.24 ± 0.05	2.24 ± 0.05	0.85 ± 0.01	0.74 ± 0.01
0.5	0.9	5.41 ± 0.05	5.06 ± 0.04	1.98 ± 0.35	1.98 ± 0.33	0.88 ± 0.01	0.73 ± 0.03
0.75	0.5	6.17 ± 0.07	5.41 ± 0.06	2.37 ± 0.02	2.37 ± 0.02	0.79 ± 0.02	0.72 ± 0.02
0.75	0.75	7.69 ± 0.06	6.84 ± 0.06	2.41 ± 0.07	2.41 ± 0.07	0.82 ± 0.01	0.71 ± 0.02
0.75	0.9	8.54 ± 0.13	7.96 ± 0.11	2.27 ± 0.49	2.27 ± 0.47	0.86 ± 0.01	0.71 ± 0.04
0.9	0.5	7.31 ± 0.07	6.40 ± 0.07	2.64 ± 0.02	2.63 ± 0.02	0.75 ± 0.02	0.71 ± 0.01
0.9	0.75	9.66 ± 0.15	8.57 ± 0.14	2.95 ± 0.10	2.94 ± 0.10	0.78 ± 0.02	0.72 ± 0.01
0.9	0.9	11.04 ± 0.09	10.28 ± 0.08	2.98 ± 0.57	2.97 ± 0.56	0.82 ± 0.01	0.70 ± 0.02

From these results, we see that the probabilistic clique cuts are able to greatly improve the bound provided by the relaxation (7), particularly so when the expected density is high, where they nearly cut the ratio by 75%. However, their overall performance is better when the expected density is low, which is in line with the performance of linear relaxations of the deterministic node packing problem. Path and cycle cuts do not decrease the bound as much as clique cuts in all of the dense instances. Possible improvements could come from increasing the number or length of the path and cycle cuts, but there is a corresponding computation time increase, and our preliminary

experiments did not show better bounds with longer paths.

Surprisingly, the MEWH performs relatively well across the board, consistently achieving a ratio between 0.75 and 0.88, with the best performance coming when $p_1 = 0.5$ and $p_2 = 0.9$, i.e. when the topology has relatively low density but edges are very likely to realize. The DCH performs worse than the MEWH across all instances, achieving ratios between 0.70 and 0.74. Its performance is fairly consistent among all the instances, in absolute terms performing best when the expected density is low. Interestingly, the DCH maintains its performance better than the MEWH as the expected density increases. This is likely a result of the probabilistic clique constraints increasing in strength as the expected density rises.

Table 2 below summarizes results for the sparse instances, following a similar format to Table 1; the Appendix again includes full results. We observe that all the tested bounds perform better than they did in dense instances. The base relaxation (7) has at most a 9% gap and improves with p; this is not surprising, because when $p = 1$, (7) coincides with the linear relaxation of the deterministic model's edge formulation, which is tight for trees. The path cuts are able to reduce the gap to nearly zero in all cases, improving again as p increases. Finally, the MEWH also performs extremely well here, with a gap under 1% in all cases.

Table 2: Geometric mean and standard deviation of sparse instance results as ratios of the HSO bound.

p	(7)	(7) + Path	MEWH
0.5	1.091 ± 0.008	1.011 ± 0.004	0.992 ± 0.002
0.75	1.071 ± 0.009	1.005 ± 0.003	0.993 ± 0.003
0.9	1.033 ± 0.004	1.001 ± 0.001	0.992 ± 0.004

5.4 Impact of Non-Uniform Probabilities

In the previous experiments, all instances have a uniform non-zero probability. In particular, for the dense instances we obtained the best bound with the probabilistic clique inequalities (9); however, if the non-zero probabilities are not uniform, we must use a relaxed version of (9) with the minimum probability of all pairs in the clique. We now explore the impact of non-uniform probabilities on the quality of this upper bound.

As in the previous section, we consider graph topologies generated using $G(100, p_1)$ for $p_1 \in \{0.5, 0.75, 0.9\}$. We now choose edge probabilities uniformly at random from the intervals $[0.75 - \delta, 0.75 + \delta]$ for $\delta \in \{0, 0.05, 0.15, 0.25\}$; for $\delta = 0$ this is equivalent to our prior experiment with $p_2 = 0.75$. For each instance we compare the upper bound (UB) provided by (7) strengthened

with path, cycle, and clique constraints against the HSO bound. The chart in Figure 1 plots the ratio UB/HSO as a function of the half-width of the edge probability intervals.

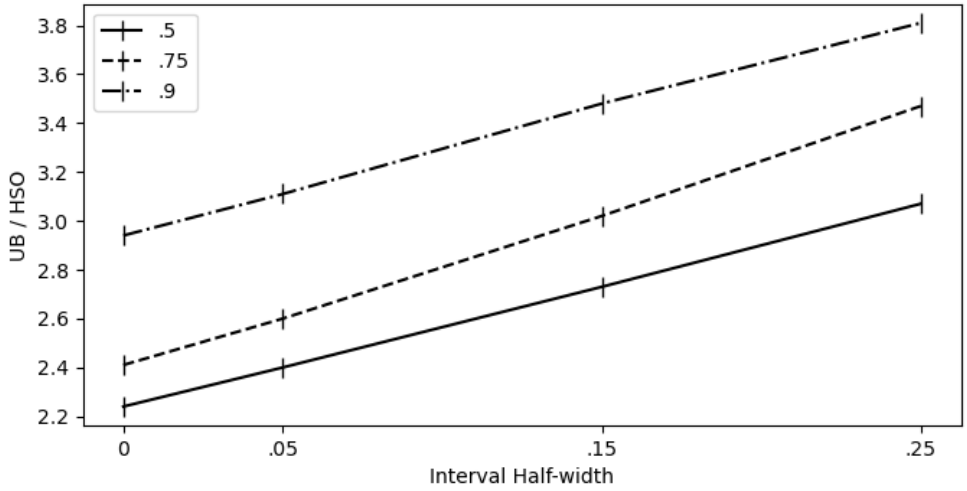

Figure 1: Effect of edge probability variation on bound quality.

As expected, the bound's quality is negatively affected by increasing variation in the edge probabilities; increasing the half-width from 0 to 0.25 increases the ratio by approximately 30% to 45% in relative terms for all three values of p_1.

6 Conclusion

We introduced a dynamic model of the weighted node packing problem. Our model generalizes the deterministic node packing problem by introducing both elements of uncertainty and dynamic decision making. Specifically, we provide a model in which edge information is revealed dynamically during the node packing process. We formulate the DNP as a Markov decision process and focus on studying the corresponding polytope of achievable probabilities. Our study yields the achievable probability polytope for two important structures; for cliques, we gave an explicit representation and showed that it is a submodular polyhedron, while for paths and cycles we implicitly characterized the polytope via a separation routine using a cut-generating LP. Our inequalities are instrumental in reducing the upper bound of the egde-based linear relaxation; for dense instances, clique cuts can reduce the gap by nearly 75%, while in sparse instances based on trees, our path inequalities suffice for near-optimality.

Nevertheless, there is a significant gap left to close, especially in denser instances. For example, a natural question is how to extend our results on probabilistic cliques with non-uniform edge probabilities beyond the case of triangles, although our preliminary results in this direction show that even for probabilistic analogues of K_4 the number of facets is very large, and their different

structures are quite varied. Another promising direction is to study probabilistic analogues of other simple graph structures, such as claws, which are known to be important in the deterministic case.

Acknowledgements

The authors' work was partially supported by the U.S. National Science Foundation via grant CMMI-1552479. Christopher Muir's work was also supported via a U.S. NSF Graduate Research Fellowship. The authors would like to thank the review team for their valuable feedback, which helped strengthen the manuscript.

References

[1] Daniel Adelman. Price-directed replenishment of subsets: Methodology and its application to inventory routing. *Manufacturing & Service Operations Management*, 5(4):348–371, 2003.

[2] Shabbir Ahmed and Alper Atamtürk. Maximizing a class of submodular utility functions. *Mathematical Programming*, 128(1):149–169, Jun 2011.

[3] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004.

[4] Stefano Basagni. Finding a maximal weighted independent set in wireless networks. *Telecommunication Systems*, 18(1-3):155–168, 2001.

[5] Dimitris Bertsimas and José Niño-Mora. Conservation laws, extended polymatroids and multiarmed bandit problems; a polyhedral approach to indexable systems. *Mathematics of Operations Research*, 21(2):257–306, 1996.

[6] Daniel Blado, Weihong Hu, and Alejandro Toriello. Semi-infinite relaxations for the dynamic knapsack problem with stochastic item sizes. *SIAM Journal on Optimization*, 26(3):1625–1648, 2016.

[7] Daniel Blado and Alejandro Toriello. Relaxation analysis for the dynamic knapsack problem with stochastic item sizes. *SIAM Journal on Optimization*, 29(1):1–30, 2019.

[8] Daniel Blado and Alejandro Toriello. A column and constraint generation algorithm for the dynamic knapsack problem with stochastic item sizes. *Mathematical Programming Computation*, 2020. Forthcoming.

[9] Joan Boyar, Lene M Favrholdt, Christian Kudahl, and Jesper W Mikkelsen. Advice complexity for a class of online problems. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[10] William Brendel, Mohamed Amer, and Sinisa Todorovic. Multiobject tracking as maximum weight independent set. In *CVPR 2011*, pages 1273–1280. IEEE, 2011.

[11] Lázaro Cánovas, Mercedes Landete, and Alfredo Marín. New facets for the set packing polytope. *Operations Research Letters*, 27(4):153–161, 2000.

[12] Eddie Cheng and William H Cunningham. Wheel inequalities for stable set polytopes. *Mathematical programming*, 77(2):389–421, 1997.

[13] Edward G Coffman Jr and Isi Mitrani. A characterization of waiting time performance realizable by single-server queues. *Operations Research*, 28(3-part-ii):810–821, 1980.

[14] Daniela Pucci De Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

[15] Brian C Dean, Michel X Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 395–404. Society for Industrial and Applied Mathematics, 2005.

[16] Brian C Dean, Michel X Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

[17] Cyrus Derman, Gerald J Lieberman, and Sheldon M Ross. A renewal decision problem. *Management Science*, 24(5):554–561, 1978.

[18] Stefan Dobrev, Rastislav Královič, and Richard Královič. Independent set with advice: the impact of graph knowledge. In *International Workshop on Approximation and Online Algorithms*, pages 2–15. Springer, 2012.

[19] Stefan Dobrev, Rastislav Královič, and Richard Královič. Advice complexity of maximum independent set in sparse and bipartite graphs. *Theory of Computing Systems*, 56(1):197–219, 2015.

[20] Uriel Feige and Robert Krauthgamer. The probable value of the Lovász-Schrijver relaxations for maximum independent set. *SIAM Journal on Computing*, 32:2003, 2003.

[21] Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online independent set beyond the worst-case: Secretaries, prophets, and periods. In *International Colloquium on Automata, Languages, and Programming*, pages 508–519. Springer, 2014.

[22] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[23] Magnús M. Halldórsson, Kazuo Iwama, Shuichi Miyazaki, and Shiro Taketomi. Online independent sets. *Theoretical Computer Science*, 289(2):953 – 962, 2002. Computing and Combinatorics.

[24] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182(1):105–142, 1999.

[25] Ferenc Juhász. The asymptotic behaviour of Lovász' ϑ function for random graphs. *Combinatorica*, 2(2):153–155, Jun 1982.

[26] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[27] Mikhail Y Kovalyov, CT Ng, and TC Edwin Cheng. Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European journal of operational research*, 178(2):331–342, 2007.

[28] Irwin Kra and Santiago R Simanca. On circulant matrices. *Notices of the AMS*, 59(3):368–377, 2012.

[29] Ojeong Kwon, Kyungsik Lee, and Sungsoo Park. Targeting and scheduling problem for field artillery. *Computers & industrial engineering*, 33(3-4):693–696, 1997.

[30] Bing-Hong Liu, Ngoc-Tu Nguyen, Van-Trung Pham, and Yu-Huan Yeh. A maximum-weight-independent-set-based algorithm for reader-coverage collision avoidance arrangement in rfid networks. *IEEE Sensors Journal*, 16(5):1342–1350, 2015.

[31] László Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

[32] Evgeny Maslov, Mikhail Batsyn, and Panos M Pardalos. Speeding up branch and bound algorithms for solving the maximum clique problem. *Journal of Global Optimization*, 59(1):1–21, 2014.

[33] Aristide Mingozzi, Vittorio Maniezzo, Salvatore Ricciardelli, and Lucio Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management science*, 44(5):714–729, 1998.

[34] George J Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*, 28(3):284–304, 1980.

[35] George L Nemhauser and Leslie Earl Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6(1):48–61, 1974.

[36] George L Nemhauser and Leslie Earl Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

[37] Manfred W Padberg. On the facial structure of set packing polyhedra. *Mathematical programming*, 5(1):199–215, 1973.

[38] Panos M. Pardalos and Jue Xue. The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328, Apr 1994.

[39] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[40] Michael Pinedo. Stochastic scheduling with release dates and due dates. *Operations Research*, 31(3):559–572, 1983.

[41] Patrick Prosser. Exact algorithms for maximum clique: A computational study. *Algorithms*, 5, 07 2012.

[42] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[43] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003.

[44] Neil JA Sloane. On single-deletion-correcting codes. *Codes and designs*, 10:273–291, 2000.

[45] Alejandro Toriello, William B Haskell, and Michael Poremba. A dynamic traveling salesman problem with stochastic arc costs. *Operations Research*, 62(5):1107–1125, 2014.

[46] Alfredo Torrico, Shabbir Ahmed, and Alejandro Toriello. A polyhedral approach to online bipartite matching. *Mathematical Programming*, 172(1-2):443–465, 2018.

[47] Leslie E Trotter Jr. A class of facet producing graphs for vertex packing polyhedra. *Discrete Mathematics*, 12(4):373–388, 1975.

[48] R. R. Vemuganti. *Applications of Set Covering, Set Packing and Set Partitioning Models: A Survey*, pages 573–746. Springer US, Boston, MA, 1998.

[49] J. Walteros and A. Buchanan. Why is maximum clique often easy in practice? *Operations Research*, 2019. Forthcoming.

[50] Hamish Waterer, E L Johnson, Paolo Nobili, and Martin WP Savelsbergh. The relation of time indexed formulations of single machine scheduling problems to the node packing problem. *Mathematical Programming*, 93(3):477–494, 2002.

[51] Qinghua Wu and Jin-Kao Hao. A review on algorithms for maximum clique problems. *European Journal of Operational Research*, 242(3):693–709, 2015.

[52] Peter J Zwaneveld, Leo G Kroon, and Stan PM Van Hoesel. Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128(1):14–33, 2001.

A Appendix: Experiment Data

Table 3: Raw data from dense instance experiments.

p_2	p_2	Ins.	7	7 + P/C	7 + Clq.	7 + P/C + Clq.	MEWH	DCH	HSO
0.5	0.5	1	75	66.20	38.58	38.58	14.07	12.49	16.86
0.5	0.5	2	75	66.30	38.13	38.13	13.8	12.58	16.76
0.5	0.5	3	75	66.35	38.44	38.44	14.12	12.58	16.77
0.5	0.5	4	75	66.07	38.41	38.41	13.65	12.56	16.77
0.5	0.5	5	75	66.21	39.14	39.14	14.23	12.83	17.3

0.5	0.5	6	75	66.21	38.81	38.81	14.02	12.62	17.04
0.5	0.5	7	75	66.21	38.24	38.24	13.47	12.03	16.65
0.5	0.5	8	75	66.01	38.66	38.66	14.04	12.59	16.84
0.5	0.5	9	75	66.07	38.47	38.47	13.73	12.51	16.75
0.5	0.5	10	75	66.19	38.46	38.46	13.94	12.2	16.75
0.5	0.75	1	62.5	55.80	27.05	27.05	10.28	9.34	12.15
0.5	0.75	2	62.5	56.09	26.68	26.68	10.23	8.92	12.04
0.5	0.75	3	62.5	55.88	26.93	26.93	10.36	9.01	12.13
0.5	0.75	4	62.5	55.92	26.91	26.91	9.99	8.71	11.95
0.5	0.75	5	62.5	55.97	27.50	27.50	10.44	8.99	12.29
0.5	0.75	6	62.5	55.87	27.24	27.24	10.29	9.13	12.23
0.5	0.75	7	62.5	55.72	26.77	26.77	9.92	8.68	11.77
0.5	0.75	8	62.5	55.91	27.12	27.12	10.8	8.73	12.14
0.5	0.75	9	62.5	55.94	26.95	26.95	10.08	8.86	12.06
0.5	0.75	10	62.5	55.91	26.95	26.95	10.06	8.58	11.89
0.5	0.9	1	55	51.49	20.19	20.19	9	7.37	10.19
0.5	0.9	2	55	51.47	19.90	19.90	8.85	7.98	10.23
0.5	0.9	3	55	51.52	20.10	20.10	8.98	7.78	10.08
0.5	0.9	4	55	51.45	20.09	20.09	8.7	7.56	10.07
0.5	0.9	5	55	51.51	20.55	20.55	9.18	7.84	10.39
0.5	0.9	6	55	51.51	20.35	20.35	9.03	7.37	10.22
0.5	0.9	7	55	51.46	19.97	19.97	8.57	7.01	10
0.5	0.9	8	55	51.35	20.25	20.25	9.42	7.12	10.37
0.5	0.9	9	55	51.44	20.11	20.11	8.83	7.48	10.14
0.5	0.9	10	55	51.52	20.11	20.11	8.48	7.28	10.01
0.75	0.5	1	75	65.76	28.50	28.50	9.56	8.73	12.22
0.75	0.5	2	75	65.87	28.34	28.34	9.55	8.67	12.03
0.75	0.5	3	75	65.72	29.71	29.70	9.78	8.64	12.16
0.75	0.5	4	75	65.75	29.52	29.49	9.65	8.75	12.24
0.75	0.5	5	75	65.70	28.77	28.77	9.59	8.74	12.08
0.75	0.5	6	75	65.80	28.22	28.22	9.74	8.53	12.24
0.75	0.5	7	75	65.74	29.20	29.20	9.62	8.58	12.06
0.75	0.5	8	75	65.85	28.93	28.93	9.7	8.88	12.17
0.75	0.5	9	75	65.67	28.00	28.00	9.54	8.99	12.22
0.75	0.5	10	75	65.72	29.57	29.57	9.64	9.2	12.23
0.75	0.75	1	62.5	55.61	19.22	19.22	6.77	6.05	8.2
0.75	0.75	2	62.5	55.58	19.21	19.20	6.53	5.78	8.01
0.75	0.75	3	62.5	55.60	20.35	20.35	6.73	5.51	8.07
0.75	0.75	4	62.5	55.67	20.22	20.22	6.87	5.65	8.13
0.75	0.75	5	62.5	55.68	19.60	19.60	6.71	5.94	8.07
0.75	0.75	6	62.5	55.60	18.99	18.99	6.71	5.7	8.12
0.75	0.75	7	62.5	55.55	19.87	19.87	6.57	5.66	8.14
0.75	0.75	8	62.5	55.58	19.55	19.55	6.75	5.7	8.12
0.75	0.75	9	62.5	55.61	18.80	18.80	6.7	5.8	8.19
0.75	0.75	10	62.5	55.52	20.13	20.13	6.68	5.76	8.22
0.75	0.9	1	55	51.37	14.24	14.24	5.51	4.66	6.54
0.75	0.9	2	55	51.27	14.34	14.34	5.57	4.44	6.27
0.75	0.9	3	55	51.27	15.31	15.31	5.48	4.65	6.32
0.75	0.9	4	55	51.39	15.20	15.20	5.51	4.53	6.46
0.75	0.9	5	55	51.31	14.68	14.68	5.56	4.25	6.37
0.75	0.9	6	55	51.23	14.06	14.06	5.49	4.52	6.41

0.75	0.9	7	55	51.29	14.85	14.85	5.46	4.29	6.35
0.75	0.9	8	55	51.27	14.49	14.49	5.82	5.29	6.57
0.75	0.9	9	55	51.27	13.90	13.90	5.51	4.44	6.6
0.75	0.9	10	55	51.32	15.03	15.03	5.56	4.98	6.54
0.9	0.5	1	75	65.64	25.21	25.18	7.8	7.33	10.32
0.9	0.5	2	75	65.62	24.59	24.54	7.84	7.31	10.34
0.9	0.5	3	75	65.69	23.06	23.06	7.91	7.42	10.38
0.9	0.5	4	75	65.73	28.44	28.19	7.75	7.27	10.22
0.9	0.5	5	75	65.61	25.16	25.17	7.81	7.4	10.27
0.9	0.5	6	75	65.63	27.58	27.44	7.59	7.41	10.22
0.9	0.5	7	75	65.72	30.39	30.29	7.57	7.31	10.19
0.9	0.5	8	75	65.67	31.61	31.27	7.56	7.2	10.18
0.9	0.5	9	75	65.73	23.96	23.94	7.75	7.28	10.29
0.9	0.5	10	75	65.75	32.82	32.24	7.73	7.21	10.22
0.9	0.75	1	62.5	55.49	17.65	17.65	5.09	4.54	6.44
0.9	0.75	2	62.5	55.55	16.94	16.94	5.2	4.7	6.46
0.9	0.75	3	62.5	55.47	15.59	15.59	5.15	4.72	6.65
0.9	0.75	4	62.5	55.45	20.49	20.45	5.06	4.51	6.41
0.9	0.75	5	62.5	55.46	17.55	17.47	5.14	4.63	6.47
0.9	0.75	6	62.5	55.48	19.42	19.42	4.94	4.81	6.46
0.9	0.75	7	62.5	55.44	21.80	21.75	5.05	4.63	6.49
0.9	0.75	8	62.5	55.44	23.11	22.94	4.91	4.63	6.36
0.9	0.75	9	62.5	55.43	16.38	16.38	5.24	4.57	6.58
0.9	0.75	10	62.5	55.53	24.19	23.83	5.02	4.63	6.41
0.9	0.9	1	55	51.22	13.56	13.55	4	3.39	4.97
0.9	0.9	2	55	51.18	12.92	12.92	4.03	3.44	4.94
0.9	0.9	3	55	51.25	11.61	11.61	4.07	3.37	5.04
0.9	0.9	4	55	51.18	16.18	16.13	4.03	3.5	4.94
0.9	0.9	5	55	51.20	13.49	13.49	4.09	3.52	4.99
0.9	0.9	6	55	51.22	14.95	14.95	3.95	3.59	4.99
0.9	0.9	7	55	51.19	17.17	17.08	4.17	3.6	4.99
0.9	0.9	8	55	51.20	18.58	18.43	4.01	3.55	4.94
0.9	0.9	9	55	51.21	12.38	12.38	4.18	3.58	5.05
0.9	0.9	10	55	51.17	19.68	19.54	4.12	3.51	4.95

Table 4: Raw data from sparse instance experiments.

p	Ins.	(7)	(7) + Path	MEWH	HSO
0.5	1	518.50	480.53	471.54	474.24
0.5	2	555.00	521.19	511.74	515.01
0.5	3	495.00	458.11	448.71	453.85
0.5	4	505.50	469.27	460.94	463.93
0.5	5	573.50	531.09	520.48	522.86
0.5	6	491.50	454.67	446.13	449.05
0.5	7	473.50	441.21	435.79	439.71
0.5	8	598.00	550.91	537.61	541.63
0.5	9	545.50	503.89	496.45	499.94
0.5	10	515.00	474.96	465.97	471.38
0.75	1	463.75	436.15	432.05	435.26
0.75	2	436.50	414.37	409.50	413.03
0.75	3	489.00	460.26	455.15	457.07

0.75	4	495.00	463.77	459.04	461.25
0.75	5	474.25	444.95	440.10	443.03
0.75	6	476.25	444.19	437.16	442.31
0.75	7	467.75	439.44	435.08	436.94
0.75	8	430.00	406.07	401.81	405.95
0.75	9	483.00	452.44	445.14	449.15
0.75	10	508.75	471.93	463.81	467.32
0.9	1	436.60	423.04	419.01	423.12
0.9	2	442.90	428.75	424.86	427.94
0.9	3	430.30	416.50	414.31	416.96
0.9	4	435.40	419.77	416.19	419.08
0.9	5	469.40	457.15	453.48	456.88
0.9	6	405.00	394.64	392.01	394.70
0.9	7	400.10	387.28	384.07	386.64
0.9	8	404.60	390.60	382.99	389.90
0.9	9	459.60	445.66	443.53	444.45
0.9	10	477.80	464.98	461.41	464.38

Table 5: Raw data from non-uniform probability experiments.

p_1	Inst.	Half-width	7 + P/C + Clq.	HSO
0.5	1	0	27.05	12.15
0.5	2	0	26.68	12.04
0.5	3	0	26.93	12.13
0.5	4	0	26.91	11.95
0.5	5	0	27.50	12.29
0.5	6	0	27.24	12.23
0.5	7	0	26.77	11.77
0.5	8	0	27.12	12.14
0.5	9	0	26.95	12.06
0.5	10	0	26.95	11.89
0.5	1	0.05	29.05	12.17
0.5	2	0.05	28.65	12
0.5	3	0.05	28.96	12.2
0.5	4	0.05	28.92	12.04
0.5	5	0.05	29.48	12.34
0.5	6	0.05	29.23	12.3
0.5	7	0.05	28.79	11.79
0.5	8	0.05	29.13	12.12
0.5	9	0.05	28.95	12.1
0.5	10	0.05	28.94	11.95
0.5	1	0.15	33.10	12.17
0.5	2	0.15	32.70	12.16
0.5	3	0.15	33.03	12.16
0.5	4	0.15	32.99	12.03
0.5	5	0.15	33.40	12.31
0.5	6	0.15	33.24	12.27
0.5	7	0.15	32.86	11.77
0.5	8	0.15	33.27	12.08
0.5	9	0.15	32.94	12.14
0.5	10	0.15	32.98	11.91

0.5	1	0.25	37.27	12.11
0.5	2	0.25	36.97	12.11
0.5	3	0.25	37.13	12.25
0.5	4	0.25	37.18	11.93
0.5	5	0.25	37.34	12.35
0.5	6	0.25	37.26	12.26
0.5	7	0.25	36.97	11.83
0.5	8	0.25	37.61	12.14
0.5	9	0.25	36.98	12.07
0.5	10	0.25	37.06	11.95
0.75	1	0	19.22	8.2
0.75	2	0	19.20	8.01
0.75	3	0	20.35	8.07
0.75	4	0	20.22	8.13
0.75	5	0	19.60	8.07
0.75	6	0	18.99	8.12
0.75	7	0	19.87	8.14
0.75	8	0	19.55	8.12
0.75	9	0	18.80	8.19
0.75	10	0	20.13	8.22
0.75	1	0.05	20.82	8.18
0.75	2	0.05	20.75	8.07
0.75	3	0.05	21.93	8.13
0.75	4	0.05	21.79	8.14
0.75	5	0.05	21.20	8.06
0.75	6	0.05	20.59	8.14
0.75	7	0.05	21.47	8.17
0.75	8	0.05	21.20	8.12
0.75	9	0.05	20.41	8.23
0.75	10	0.05	21.79	8.2
0.75	1	0.15	24.21	8.22
0.75	2	0.15	24.06	8.13
0.75	3	0.15	25.24	8.12
0.75	4	0.15	25.04	8.12
0.75	5	0.15	24.46	8.04
0.75	6	0.15	23.98	8.17
0.75	7	0.15	24.81	8.18
0.75	8	0.15	24.70	8.12
0.75	9	0.15	23.84	8.14
0.75	10	0.15	25.19	8.18
0.75	1	0.25	28.08	8.3
0.75	2	0.25	27.76	8.08
0.75	3	0.25	28.85	8.1
0.75	4	0.25	28.55	8.16
0.75	5	0.25	28.01	8.1
0.75	6	0.25	27.79	8.16
0.75	7	0.25	28.58	8.09
0.75	8	0.25	28.64	8.14
0.75	9	0.25	27.69	8.19
0.75	10	0.25	28.93	8.23
0.9	1	0	17.65	6.44

0.9	2	0	16.94	6.46
0.9	3	0	15.59	6.65
0.9	4	0	20.45	6.41
0.9	5	0	17.47	6.47
0.9	6	0	19.42	6.46
0.9	7	0	21.75	6.49
0.9	8	0	22.94	6.36
0.9	9	0	16.38	6.58
0.9	10	0	23.83	6.41
0.9	1	0.05	18.77	6.42
0.9	2	0.05	18.07	6.54
0.9	3	0.05	16.84	6.65
0.9	4	0.05	21.48	6.49
0.9	5	0.05	18.74	6.54
0.9	6	0.05	20.77	6.53
0.9	7	0.05	22.81	6.62
0.9	8	0.05	23.92	6.39
0.9	9	0.05	17.69	6.56
0.9	10	0.05	25.03	6.35
0.9	1	0.15	21.06	6.46
0.9	2	0.15	20.42	6.42
0.9	3	0.15	19.50	6.63
0.9	4	0.15	23.49	6.41
0.9	5	0.15	21.19	6.45
0.9	6	0.15	22.99	6.46
0.9	7	0.15	25.06	6.49
0.9	8	0.15	26.04	6.36
0.9	9	0.15	20.21	6.65
0.9	10	0.15	26.61	6.39
0.9	1	0.25	23.39	6.48
0.9	2	0.25	22.98	6.43
0.9	3	0.25	22.43	6.8
0.9	4	0.25	25.78	6.47
0.9	5	0.25	23.81	6.49
0.9	6	0.25	25.33	6.54
0.9	7	0.25	27.04	6.5
0.9	8	0.25	27.50	6.4
0.9	9	0.25	22.89	6.68
0.9	10	0.25	28.08	6.49