Submitted to *INFORMS Journal on Optimization*
manuscript IJO-2019-10-027

Tree Bounds for Sums of Bernoulli Random Variables: A Linear Optimization Approach

Divya Padmanabhan
Engineering Systems Design, Singapore University of Technology and Design, divya_padmanabhan@sutd.edu.sg
Karthik Natarajan
Engineering Systems Design, Singapore University of Technology and Design, karthik_natarajan@sutd.edu.sg

We study the problem of computing the tightest upper and lower bounds on the probability that the sum of n dependent Bernoulli random variables exceeds an integer k. Under knowledge of all pairs of bivariate distributions denoted by a complete graph, the bounds are NP-hard to compute. When the bivariate distributions are specified on a tree graph, we show that tight bounds are computable in polynomial time using a compact linear program. These bounds provide robust probability estimates when the assumption of conditional independence in a tree structured graphical model is violated. We demonstrate, through numericals, the computational advantage of our compact linear program over alternate approaches. A comparison of bounds under various knowledge assumptions such as univariate information and conditional independence is provided. An application is illustrated in the context of Chow Liu trees, wherein our bounds distinguish between various trees which encode the maximum possible mutual information.

Key words: probability bounds, trees, linear optimization
History: This paper was first submitted in October 2019 and revised in April 2020.

1. Introduction

Analysis of the sums of Bernoulli random variables have received much attention among researchers in probability, computer science, optimization and engineering due to its wide applicability. For example, an insurer in risk management is interested in estimating the probability that the number of defaults among n claims is k or more [Wang, 1998]. In the context of reliability, the probability that at least k of n subsystems is functional aids in estimating the probability that the entire system is functional [Boland and Proschan, 1983].

Our interest is in the setting where bivariate dependence information among the Bernoulli random variables is available. Formally, denote by $\tilde{\mathbf{c}}$, an n-dimensional Bernoulli random vector.

Associated with this random vector is a graph $G=(V,E)$ where V is the set of n vertices denoting the random variables and E is the set of edges denoting the pairs of random variables for which bivariate information is specified. The univariate distributions are denoted as $\mathbb{P}(\tilde{c}_i = r_i)$ for $i \in V,\ r_i \in \{0,1\}$ and the bivariate distributions are denoted as $\mathbb{P}(\tilde{c}_i = r_i, \tilde{c}_j = r_j)$ for $(i,j) \in E$, $r_i \in \{0,1\}, r_j \in \{0,1\}$. Let Θ denote the set of distributions as follows:

$$\Theta = \left\{\theta \in \mathbb{P}(\{0,1\}^n) : \mathbb{P}_\theta\left(\tilde{c}_i = 1, \tilde{c}_j = 1\right) = p_{ij} \text{ for } (i,j) \in E,\ \mathbb{P}_\theta\left(\tilde{c}_i = 1\right) = p_i \text{ for } i \in V\right\},$$

where $\mathbb{P}(\{0,1\}^n)$ is the set of probability distributions of n dimensional Bernoulli random vectors and $\mathbb{P}_\theta\left(\tilde{c}_i = 1, \tilde{c}_j = 0\right) = p_i - p_{ij}$, $\mathbb{P}_\theta\left(\tilde{c}_i = 0, \tilde{c}_j = 1\right) = p_j - p_{ij}$ and $\mathbb{P}_\theta\left(\tilde{c}_i = 0, \tilde{c}_j = 0\right) = 1 - p_i - p_j + p_{ij}$. Define $U(k)$ and $L(k)$ as the largest and smallest possible probability that the sum of the n random variables exceeds an integer k computed over all distributions in Θ:

$$U(k) = \max_{\theta \in \Theta}\ \mathbb{P}_\theta\left(\sum_{i=1}^{n} \tilde{c}_i \geq k\right),$$
$$L(k) = \min_{\theta \in \Theta}\ \mathbb{P}_\theta\left(\sum_{i=1}^{n} \tilde{c}_i \geq k\right).$$

Unfortunately when the graph is complete with rational entries p_{ij} and p_i, verifying if the set Θ is nonempty is an NP-complete problem (Theorem 3.3 in [Pitowsky, 1991]). This implies that computing the tight bounds efficiently is highly unlikely, unless P = NP. For example, one can construct simple instances, even with $n = 3$ where the random variables are pairwise consistent (pairs of Bernoulli random variables exist) but not globally consistent (a multivariate Bernoulli random vector does not exist). An instance is $p_1 = p_2 = p_3 = 1/2$ and $p_{12} = p_{23} = p_{13} = 0$ where Θ is empty while the bivariates are pairwise consistent (see Vorobev [1962]).

1.1. Tree Graphs

A natural approach is to consider simpler graph structures where the feasibility of Θ is easy to verify. Towards, this, we consider the class of tree graphs $T=(V,E)$ which has attractive computational properties. Such graphs have been extensively studied in graphical models in computer science and machine learning ([Chow and Liu, 1968, Lauritzen, 1996, Wainwright and Jordan, 2008]) where a tree structured distribution that exploits conditional independence is used from the set Θ. The tree structure itself is known, for example, in the context of vine copulas. Vine copulas have been used as effective tools for dependence modelling in several areas such as finance [Dissmann et al., 2013] and longitudinal studies [Cooke et al., 2019]. Given a set of n random variables, a vine is an ordered set of $n-1$ trees. The first tree T_1 contains n nodes and the edges in T_1 denote the bivariate dependency of the pairs of nodes represented by the edges. In general, the edges of tree l serve as the nodes of tree $l+1$. Each edge in tree T_l contains the bivariate dependency information between two vectors of random variables each of size l, conditional on the common variables in the

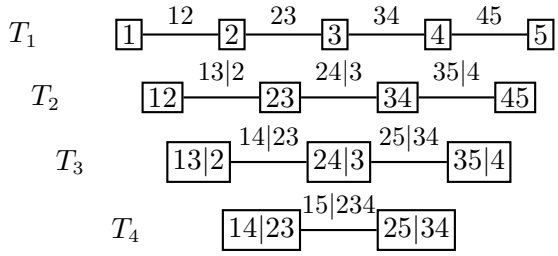


Figure 1 **A special case of a vine copula for $n=5$ random variables where every tree T_l is a series graph.**

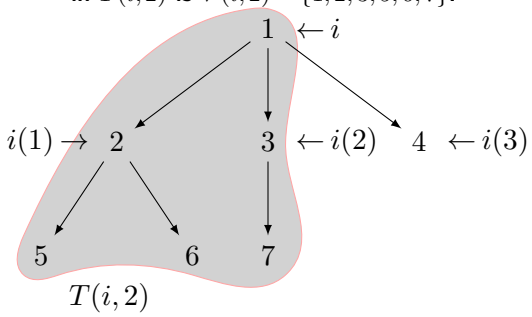
nodes connected by the edge. An example vine copula is illustrated for the case of $n=5$ random variables in Figure 1. The theory on vine copulas looks at learning the structure of the vine itself as well as estimating the corresponding probability distributions on the edges of the trees from data. Much work has been devoted to both these aspects [Brechmann and Joe, 2015, Chang et al., 2019, Dissmann et al., 2013]. There are also several packages towards this (for example, the VineCopula package in R). In general, estimating higher order trees is a difficult problem. Many works therefore resort to a truncated vine where only the probability distributions corresponding to the lower order trees (upto some level M) are learnt and all distributions in higher order trees are set to the independent distribution [Kurowicka, 2010]. Truncated vines can be viewed as sparse vines and fit the provided dataset well in many scenarios [Brechmann et al., 2012]. In particular, when $M=1$, the first tree T_1 provides the dependencies on pairs of random variables specified by the edges of the tree. It is well known that given this tree, the conditionally independent distribution maximizes the entropy. Our work aids in computing robust estimates for the given truncated vine at level 1, when no assumptions of conditional independence are made. We however do not look at estimating the vine structure itself; we prescribe using the known results on vine structure learning for this.

We now focus on a directed rooted tree T representation of the graph where node 1 is designated as the root and the arcs are directed away from the root node. Assume an arbitrary but fixed ordering of the remaining nodes. The parent of a node $i \neq 1$ is denoted as $\text{par}(i)$ and is the unique node that connects to i on the path from the root node. A child of a node i is a node for which i is the parent. A descendant of a node i refers to any of the children of i or the descendants of the children of i. A leaf node is a node with no descendants. We let d_i denote the out-degree of node i and denote the sth child of node i (as per the ordering fixed a-priori) as $i(s)$. We denote by $T(i,s)$ the sub-tree rooted at i consisting of the first s sub-trees of i where $V(i,s)$ is the set of vertices in $T(i,s)$ and $N(i,s)$ is the cardinality of this set. For ease of understanding, the notations are illustrated in Figure 2 below.

Given the bivariate distributions for this tree, a feasible distribution is given by:

$$\mathbb{P}(\tilde{\mathbf{c}}=\mathbf{r}) = \mathbb{P}(\tilde{c}_1 = r_1) \prod_{j \neq 1} \mathbb{P}(\tilde{c}_j | \tilde{c}_{\text{par}(j)} = r_{\text{par}(j)}). \tag{1}$$

Figure 2 **Suppose $n=7$ and the set of known bivariate marginal distributions corresponds to the set $E=\{(1,2),(2,5),(2,6),(3,7),(3,1),(4,1)\}$. The figure gives the corresponding directed tree with all arcs pointing away from the root node 1. The parent of nodes 5 and 6 is 2, the parent of node 7 is 3 and the parent of nodes 2, 3 and 4 is 1. The degrees of the various nodes are $d_1=3, d_2=2, d_3=1, d_4=0, d_5=0, d_6=0, d_7=0$. Let i denote the root node labelled 1. Assuming a non-decreasing order on the node labels, the three children of i are denoted as $i(1), i(2)$ and $i(3)$ (nodes 2,3 and 4 respectively). The sub-tree of i induced by the first two children is denoted by $T(i,2)$ and is shaded. The number of nodes in $T(i,2)$ is denoted by $N(i,2)$. $N(i,2)=6$ here. The set of vertices in $T(i,2)$ is $V(i,2)=\{1,2,3,5,6,7\}$.**



This distribution is based on the conditional independence assumption among the random variables in the tree which implies that for any two nodes $i \neq j$ such that $\text{par}(i) = \text{par}(j)$, we have:

$$\mathbb{P}(\tilde{c}_i = x | \tilde{c}_{\text{par}(i)}, \tilde{c}_j) = \mathbb{P}(\tilde{c}_i = x | \tilde{c}_{\text{par}(i)}). \tag{2}$$

As an example, the conditionally independent distribution for the tree shown in Figure 2 is:

$$\begin{aligned}\mathbb{P}(\tilde{\mathbf{c}} = \mathbf{r}) =& \mathbb{P}(\tilde{c}_1 = r_1)\mathbb{P}(\tilde{c}_2 = r_2|\tilde{c}_1 = r_1)\mathbb{P}(\tilde{c}_3 = r_3|\tilde{c}_1 = r_1)\mathbb{P}(\tilde{c}_4 = r_4|\tilde{c}_1 = r_1) \\ & \times \mathbb{P}(\tilde{c}_5 = r_5|\tilde{c}_2 = r_2)\mathbb{P}(\tilde{c}_6 = r_6|\tilde{c}_2 = r_2)\mathbb{P}(\tilde{c}_7 = r_7|\tilde{c}_3 = r_3)\end{aligned} \tag{3}$$

Conditionally independent distributions are commonly used in graphical models. Given a tree structured graphical model, many of the inference problems such as estimating the marginal distribution over a subset of random variables, computing the mode of the underlying distribution (see [Lauritzen, 1996, Wainwright and Jordan, 2008]) or estimating the probability that sum of the random variables exceed k is easy. However, much lesser is discussed in the literature on what happens when the assumption of conditional independence is violated. In this paper, we use the tree structure of the graph as a certificate that the set Θ is nonempty and evaluate for the distributions in Θ that are extremal and attain the bounds $U(k)$ and $L(k)$. As our numerical results demonstrate, these bounds can in some cases be significantly different from the probability under conditionally independent distribution. Thus the bounds in the paper can provide robust estimates on probabilities when the conditional independence assumption on the underlying structured graphical models is violated. A similar problem was recently studied by Dhara et al. [2019] where tight tree bounds were proposed for the expectation of a sum of discrete random variables beyond a threshold using linear optimization. In contrast to their work, our focus in this paper is on probability bounds which requires the use of different proof techniques.

1.2. Related Results

When only univariate probabilities p_i are known for Bernoulli random variables, the tight upper bound on $\mathbb{P}\left(\sum_{i=1}^n \tilde{c}_i \geq k\right)$ was derived by [Morgenstern, 1980, Ruger, 1978] as follows:

$$U_{uv}(k) = \min\left(\left(\min_{t\in\{0,\ldots,k-1\}} \sum_{i=1}^{n-t} \frac{p_{(i)}}{k-t}\right), 1\right) \tag{4}$$

where $p_{(1)} \leq p_{(2)} \leq \ldots \leq p_{(n)}$ are the order statistics of the probabilities p_i. When instead of exact bivariate probabilities, only lower bounds on the bivariate probabilities (and in general m-variate probabilities) are known, a tight upper bound on the tail probability is computed in polynomial time for $k = 1$ (union of events) in Boros et al. [2014]. In this scenario, verifying the existence of a feasible distribution for m=2 is, in particular, possible in polynomial time (see Chapter 8 of Bertsimas and Tsitsiklis [1997]). When the exact probabilities p_{ij} on all edges of a graph are known, the tight bound is obtained as a solution to an exponential sized linear programming formulation as discussed in [Hailperin, 1965, Kounias and Marin, 1976, Prékopa et al., 1997]. Hunter [1976] and Worsley [1982] proposed an upper bound for $\mathbb{P}(\sum_{i=1}^n c_i \geq 1)$ in terms of the total weight of a maximum spanning tree on a complete graph of n vertices, with the weight of edge (i,j) taken as the probability p_{ij}. Their proposed upper bound is $\sum_{i=1}^n p_i - \max_T \sum_{(i,j)\in T} p_{ij}$, where the maximum is computed over all possible trees T. In the specific case where the bivariate probabilities are given as $p_{ij} = 0$ for all edges (i,j) not in a tree T, Kounias [1968] show that $\sum_{i=1}^n p_i - \sum_{(i,j)\in T} p_{ij}$ is a tight upper bound for $k = 1$. Extensions of the approach to higher order information have been considered in Bukszár and Szántai [2002], Tomescu [1986]. For tree structured bivariate information, Rüschendorf [1991] proposed a conditioning method for series and star graphs. Embrechts and Puccetti [2010] also proposed upper bounds building on these results. These bounds are tight in very special cases and are in general not tight.

Under knowledge of aggregate information such as the binomial moments $S1 = \sum_{i=1}^n p_i$ and $S2 = \sum_{i=1}^n \sum_{i=1}^n p_{ij}$, several works including Boros and Prekopa [1989], Dawson and Sankoff [1967], Kwerel [1975], Prékopa [1988], Prékopa [1990a,b], Yang et al. [2016] and references therein have computed upper and lower bounds for $\mathbb{P}(\sum_{i=1}^n \tilde{c}_i \geq k)$. The problem has also been studied under the scenario of known partial aggregate information (knowledge of $\sum_{j=1}^n p_{ij}$ for each i) in Prékopa and Gao [2005], Qiu et al. [2016], Yang et al. [2016] with polynomial time solvable linear programming relaxations in Qiu et al. [2016]. All these papers work in the setting of aggregate information, whereas in our work, we assume knowledge of disaggregated information, in the sense that we are provided each of the individual probabilities $\mathbb{P}(\tilde{c}_i = 1, \tilde{c}_j = 1)$ and $\mathbb{P}(\tilde{c}_i = 1)$, which together uniquely encode the bivariate distribution between the pair $(\tilde{c}_i, \tilde{c}_j)$. We provide a snapshot of the key results in the space of disaggregated information in Table 1.

A closely related body of work is the theory on probabilistic programming and chance constraints [Prékopa, 2003, Saxena et al., 2009]. Traditionally probabilistic programming deals with optimizing a function with upper/lower bound constraints on the probability of related uncertain events where the probability distribution governing the underlying uncertain quantity is known. Moving on to the realm of distributionally robust chance constraints, Ahmed and Papageorgiou [2013] look at solving the two stage problem, $\max_{\mathbf{x}\in\mathcal{X}\subseteq\{0,1\}^n} \min_{\theta\in\Theta_{\text{full}}} \mathbb{P}_\theta(\tilde{\mathbf{c}}'\mathbf{x}\geq 1)$ for a Bernoulli vector $\tilde{\mathbf{c}}$. For the particular case of $\mathbf{x}=\mathbf{1}_n$, the inner minimization corresponds to $L(k)$, for the special case of $k=1$. However their ambiguity set Θ_{full} assumes knowledge of all pairs of bivariate marginals and they propose a mixed-integer linear program implicitly assuming a non-empty Θ_{full}. Our model differs in the following ways: (1) we only assume bivariate information on a tree structure (2) k can take any integer value (3) we propose a polynomial sized linear program.

1.3. Overview of Approach

We consider Hailperin [1965]'s exponential sized linear program to compute $U(k)$ for a given graph:

$$\begin{aligned}
\max_{\theta} \quad & \sum_{\mathbf{c}\in\{0,1\}^n} \theta(\mathbf{c})\ \mathbb{1}\{\textstyle\sum_i c_i \geq k\} \\
\text{s.t.} \quad & \sum_{\mathbf{c}:c_i=1} \theta(\mathbf{c}) = p_i \ \text{ for } i\in V \\
& \sum_{\mathbf{c}:c_i=1,c_j=1} \theta(\mathbf{c}) = p_{ij} \ \text{ for } (i,j)\in E \\
& \sum_{\mathbf{c}\in\{0,1\}^n} \theta(\mathbf{c}) = 1 \\
& \theta(\mathbf{c})\geq 0 \quad \text{for } \mathbf{c}\in\{0,1\}^n
\end{aligned}$$

where $\mathbb{1}\{\sum_{i=1}^n c_i\geq k\}=1$ if $\sum_{i=1}^n c_i\geq k$ and 0 otherwise and $\theta(\mathbf{c})$ denotes the probability of realization $\mathbf{c}$. The first two constraints enforce the given information on the univariate and bivariate probabilities while the last two constraints ensure that θ is a valid distribution. The formulation above is exponential sized owing to number of realizations of $\tilde{\mathbf{c}}$.

Table 1 Tight upper bound on $\max\mathbb{P}(\sum_{i=1}^n \tilde{c}_i\geq k)$ for Bernoulli random variables

Univariate	Bivariate	Solution approach	Computation
p_i	Not given	Closed form bound [Morgenstern, 1980, Ruger, 1978]	Easy
p_i	Lower bounds on bivariate probabilities in a complete graph	Tight bound for $k=1$ [Boros et al., 2014]	Easy
p_i	Exact values for bivariate probabilities in a complete graph	Exponential sized linear program [Hailperin, 1965]	Hard
p_i	Exact values for bivariate probabilities in a complete graph; bivariate probabilities are 0 for edges not in a tree	Tight bound for $k=1$ [Hunter, 1976, Worsley, 1982]	Easy
p_i	Exact values for bivariate probabilities in a tree	Linear program [Current paper]	Easy

The dual to the above formulation is:

$$\min_{\lambda,\boldsymbol{\alpha},\boldsymbol{\beta}} \ \lambda + \sum_{i=1}^{n} \alpha_i p_i + \sum_{(i,j)\in E} \beta_{ij} p_{ij}$$

$$\text{s.t. } \lambda + \sum_{i=1}^{n} \alpha_i c_i + \sum_{(i,j)\in E} \beta_{ij} c_i c_j \geq 1, \quad \text{for } \mathbf{c} \in \{0,1\}^n \text{ where} \sum_{i=1}^{n} c_i \geq k \tag{5}$$

$$\lambda + \sum_{i=1}^{n} \alpha_i c_i + \sum_{(i,j)\in E} \beta_{ij} c_i c_j \geq 0 \quad \text{for } \mathbf{c} \in \{0,1\}^n \tag{6}$$

The dual has an exponential number of constraints and can be grouped into two sets of constraints. In particular, for the separation version of the above problem, given $\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta}$, we need to verify if all constraints in (5) and (6) are met, or else we need to find a violated inequality. By the equivalence of separation and optimization in Grötschel et al. [1988], the existence of a polynomial time solution to the separation version would imply the existence of a polynomial time algorithm for the optimization problem. We observe that, indeed, the separation problem is polynomial time solvable when the given graph is a tree. Row generation or cutting plane techniques [Nemhauser and Wolsey, 2014] then provide a natural scheme for solving the problem wherein one starts with a feasible set of variables, identifies a violated inequality and updates the model by including the violated constraint explicitly (referred to as the cutting plane). Then re-optimization is performed to get a new set of variable values and the above process is repeated till none of the constraints are violated. However, the process of re-optimization can be expensive when several iterations of the separation problem need to be solved. In the worst case this could lead to inclusion of an exponential number of constraints for our specific problem, leading to increased computational times. Rather than going for row generation scheme, we make use of the polynomial time algorithm for the separation problem to develop a compact linear program to compute $U(k)$. The compact linear program has the advantage that it overcomes the potential computational hurdles of row generation posed by re-optimization. While the paper will focus primarily on $U(k)$, the computation of the lower bound $L(k)$ can be performed in terms of the upper bound $U(\cdot)$ for an appropriately defined instance as we will show later.

The rest of the paper is organized as follows. In Section 2 we consider the special case of a quadratic knapsack problem with cardinality constraints on a tree graph which arises in the dual formulation and develop a compact linear program based on a set of dynamic programming recursions. Building on this, in Section 3 we propose a polynomial sized linear programming formulation to compute $U(k)$ for tree graphs. We also compare this formulation with the corresponding approach for the conditionally independent distribution in a graphical tree model. We discuss the computation of lower bounds $L(k)$, a generalization of the bounds to weighted sums of probabilities in Section 4 and provide numerical results in Section 5.

2. Special Case of Cardinality Constrained Quadratic Knapsack

Given a graph $G=(V,E)$ and the parameter vectors $\boldsymbol{\alpha},\boldsymbol{\beta}$, consider the quadratic optimization problem:

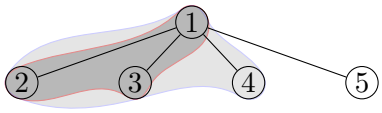
$$Q_T = \min\left\{\sum_{i=1}^{n}\alpha_i c_i + \sum_{(i,j)\in E}\beta_{ij}c_i c_j : \sum_{i=1}^{n} c_i \geq k, c_i \in \{0,1\}\forall\, i\in V\right\}, \tag{7}$$

which arises in the dual formulation. Formulation (7) is a special case of a quadratic knapsack problem with only cardinality constraints. The cost minimization version of a quadratic knapsack problem in its general form requires to find a vector $\mathbf{c}\in\{0,1\}^n$ to minimize $\sum_{i\in V}\alpha_i c_i + \sum_{(i,j)\in E}\beta_{ij}c_i c_j$ subject to a constraint $\sum_{i\in V} a_i c_i \geq k$. Each vertex can be interpreted as corresponding to an item and each item is associated with a utility a_i. The requirement is to choose a set of items so that the overall utility of the selection is at least k. To this end if an item i is selected, $c_i=1$ else $c_i=0$. Given an edge (i,j) in the graph, an additional cost of β_{ij} is incurred when both items i and j are selected, in addition to individual item costs α_i and α_j. When item i is selected but item j is not selected, a cost of α_i alone is incurred for the item. The overall goal is therefore to select a set with a total utility of at least k while minimizing the cost induced by the selected items.

Quadratic knapsack problem is NP-hard in the strong sense [Caprara et al., 1999, Fomeni and Letchford, 2014]. However for special types of graphs such as series-parallel graphs (of which trees are a special case), a pseudo-polynomial time dynamic programming algorithm of time complexity $O(nk^2)$ is available [Jr. and Woeginger, 2002].

For this problem, when the graph is a tree, a dynamic programming algorithm has been proposed in Billionnet [1992]. In this section, we will develop a linear optimization formulation for this problem that builds on dynamic programming. This linear optimization formulation will be used later in Section 3 for the computation of the probability bounds. We start with the linear programming formulations for two particular trees - the series and the star graphs and generalize the result to arbitrary trees by viewing them as a combination of several series and/or star subgraphs. Throughout the paper, we use the notation $[n]$ to denote the set $\{1,\ldots,n\}$ for any integer n, and $[i,j]$ to denote the set $\{i,i+1,\ldots,j\}$ for integers i and j.

(a) Series graph on $n=5$ nodes



(b) Star graph on $n=5$ nodes

Figure 3 Illustrative examples representing the basic components of any general tree. The shaded parts illustrate the relevant sub-trees corresponding to smaller sub-problems.

2.1. Formulation for a Series Graph

The series graph $G_{series} = (V, E_{series})$ on n nodes is a graph with the edge set $E_{series} = \{(1,2),(2,3),\ldots,(n-1,n)\}$. Such a graph contains exactly one leaf node (node n) (see Figure 3(a)). We will now propose a linear programming formulation for solving the quadratic minimization problem with a series graph:

$$\bar{Q} = \min\left\{\sum_{i=1}^{n}\alpha_i c_i + \sum_{i=1}^{n-1}\beta_{i,i+1}c_i c_{i+1} : \sum_{i=1}^{n} c_i \geq k, c_i \in \{0,1\}\forall\, i\right\} \tag{8}$$

PROPOSITION 1. *For a series graph $G_{series} = (V, E_{series})$, the optimal value $\bar{Q}$ of the quadratic knapsack problem in* (8) *can be obtained by solving the following linear program:*

$$\begin{aligned}
\bar{Q} = \max_{z,\mathbf{x}}\ & z \\
s.t.\ & z \leq x_{1,0,t}, && t \in [k, n-1] \\
& z \leq x_{1,1,t} && t \in [k, n] \\
& x_{i,0,t} \leq x_{i+1,0,t}, && t \in [0, n-i-1], i \in [1, n-1] \\
& x_{i,0,t} \leq x_{i+1,1,t}, && t \in [1, n-i], i \in [1, n-1] \\
& x_{i,1,t} \leq x_{i+1,0,t-1} + \alpha_i, && t \in [1, n-i], i \in [1, n-1] \\
& x_{i,1,t} \leq x_{i+1,1,t-1} + \alpha_i + \beta_{i,i+1}, && t \in [2, n-i+1], i \in [1, n-1] \\
& x_{n,0,0} = 0 \\
& x_{n,1,1} = \alpha_n.
\end{aligned}$$

Proof: Denote by $\bar{f}(i,y,t)$ the optimal value of formulation (8), when restricted to the sub-tree rooted at i such that t nodes are selected from this sub-tree and c_i takes a value $y \in \{0,1\}$:

$$\bar{f}(i,y,t) = \min\left\{\sum_{l\geq i}\alpha_l c_l + \sum_{l\geq i}\beta_{l,l+1}c_l c_{l+1} : \sum_{l\geq i} c_l = t, c_i = y, c_l \in \{0,1\}\forall l \geq i\right\} \tag{9}$$

If $y = 0$, the range of admissible values for t is $[0, n-i]$ while if $y = 1$, the range of admissible values for t is $[1, n-i+1]$ (see the shaded region in Figure 3(a) for the relevant sub-tree for $\bar{f}(3,\cdot,\cdot)$).

The smallest such sub-tree is rooted at node n which contains the leaf node n alone. With $c_n = y$, the only possible value of t is y which leads to the following two base-cases:

$$\bar{f}(n,0,0) = 0, \bar{f}(n,1,1) = \alpha_n. \tag{10}$$

Using the above two base cases, the optimal value for other sub-problems can be recursively computed. We will now develop the recursions for $\bar{f}(i,y,t)$ for any internal node i, for all valid values of y and t in terms of the optimal values for $\bar{f}(i+1,\cdot,\cdot)$. For any valid value of t, we have:

$$\bar{f}(i,0,t) = \min(\bar{f}(i+1,0,t), \bar{f}(i+1,1,t)) \tag{11}$$

$$\bar{f}(i,1,t) = \min(\bar{f}(i+1,0,t-1) + \alpha_i, \bar{f}(i+1,1,t-1) + \alpha_i + \beta_{i,i+1}) \tag{12}$$

When $c_i = 0$, the t nodes which take a value of 1 must all be located in the sub-tree rooted at $i+1$. The terms in the right hand side of Equation (11) deal with the case where $c_{i+1} = 0$ and $c_{i+1} = 1$ respectively. On the other hand, when $c_i = 1$, then the sub-tree rooted at $i+1$ must select $t-1$ nodes so that a total of t nodes are selected from the subtree rooted at i. Here if c_{i+1} takes a value

of 0, only an additional cost of α_i is incurred, while if $c_{i+1}=1$, then an additional cost of $\alpha_i+\beta_{i,i+1}$ is incurred. In all these cases the range of valid t varies depending on the values of c_i and c_{i+1}. For example, if $c_i=0$ and $c_{i+1}=1$, the range of permissible values of t is $[1,n-i]$ while for $c_i=1$ and $c_{i+1}=1$, the permissible range is $[2,n-i+1]$.

The optimal objective $\bar{Q}$ on the overall series graph is obtained by looking at the optimal values of $\bar{f}(\cdot,\cdot,\cdot)$ corresponding to the root node. In particular,

$$\bar{Q}=\min\left(\min_{t_1\in[k,n-1]}\bar{f}(1,0,t_1),\min_{t_2\in[k,n]}\bar{f}(1,1,t_2)\right). \tag{13}$$

While the range of permissible values of t_1 in $\bar{f}(1,0,t_1)$ is $[k,n-1]$ (as $c_1=0$ here), the permissible range of t_2 is $[k,n]$ in $\bar{f}(1,1,t_2)$. The variable $\mathbf{x}$ which is $O(n^2)$ encodes the optimal values $\bar{f}(\cdot,\cdot,\cdot)$ of all the sub problems while z encodes the optimal value $\bar{Q}$. The inequalities in the linear program arise as a consequence of linearizing the minimum operator in Equations (11) to (13). □

2.2. Formulation for a Star Graph

The star graph $G_{star}=(V,E_{star})$ on $n\geq 2$ nodes is a graph with edge set $E_{star}=\{(1,2),(1,3),\ldots,(1,n)\}$ (see Figure 3(b) for an illustration). We will now propose a linear programming formulation for solving the minimization problem:

$$Q_{star}=\min\left\{\sum_{i=1}^{n}\alpha_i c_i+\sum_{i\in[2,n]}\beta_{1i}c_1c_i:\sum_{i=1}^{n}c_i\geq k,c_i\in\{0,1\}\forall i\right\} \tag{14}$$

Proposition 2. *For a star graph $G_{star}=(V,E_{star})$, the optimal value Q_{star} of the quadratic knapsack problem in* (14) *can be obtained by solving the following linear program:*

$$\begin{aligned}
Q_{star}=\max_{z,\mathbf{x}}\ & z & \\
s.t.\ & z\leq x_{n,0,t}, & t\in[k,n-1]\\
& z\leq x_{n,1,t}, & t\in[k,n]\\
& x_{i,0,t}\leq x_{i-1,0,t}, & t\in[0,i-2],i\in[3,n]\\
& x_{i,0,t}\leq x_{i-1,0,t-1}+\alpha_i, & t\in[1,i-1],i\in[3,n]\\
& x_{i,1,t}\leq x_{i-1,1,t}, & t\in[1,i-1],i\in[3,n]\\
& x_{i,1,t}\leq x_{i-1,1,t-1}+\alpha_i+\beta_{1i}, & t\in[2,i],i\in[3,n]\\
& x_{2,0,0}=0,\ x_{2,0,1}=\alpha_2 & \\
& x_{2,1,1}=\alpha_1,\ x_{2,1,2}=\alpha_1+\alpha_2+\beta_{12}. &
\end{aligned}$$

Proof: For values of $i\geq 2$, denote by $g(i,y,t)$ the minimum value that is obtained by restricting attention to the sub-tree containing the nodes $\{1,\ldots,i\}$, such that t nodes are selected from this sub-tree and c_1 takes a value y:

$$g(i,y,t)=\min\left\{\sum_{l=1}^{i}\alpha_l c_l+\sum_{l=2}^{i}\beta_{1,l}c_1c_l:\sum_{l=1}^{i}c_l=t,c_1=y,c_l\in\{0,1\}\forall l\leq i\right\} \tag{15}$$

The region shaded in Figure 3(b) (consisting of nodes 1,2,3,4) shows the relevant tree for $g(4,\cdot,\cdot)$. If $c_1=0$, the valid values of t lie in $[0,i-1]$ while if $c_1=1$, t must lie in $[1,i]$.

We will now provide recursions to compute the values of $g(\cdot,\cdot,\cdot)$. The base conditions look at the sub-tree with exactly two nodes $\{1,2\}$ as follows:

$$g(2,0,0)=0, g(2,0,1)=\alpha_2 \tag{16}$$

$$g(2,1,1)=\alpha_1, g(2,1,2)=\alpha_1+\alpha_2+\beta_{12} \tag{17}$$

Equation (16) deals with the case where $c_1=0$ and the possible value of t is either 0 or 1, depending on the value of c_2. If $t=0$, it must be that $c_2=0$ in which case, no cost is incurred as none of the nodes are selected. If $t=1$, then the only possibility is $c_2=1$ and this brings in a cost of α_2. A similar approach can be used to consider the case with $c_1=1$ (Equation (17)) where the two possibilities are $t=1$ and $t=2$. If $t=1$, it must be that $c_2=0$ and therefore the cost incurred is only α_1 while if $t=2$, then it must be that $c_2=1$ and therefore the cost incurred is $\alpha_1+\alpha_2+\beta_{12}$.

Given these base conditions, we are now ready to compute the value of the function $g(i,\cdot,\cdot)$, for $i\geq 3$, in terms of the value corresponding to smaller sub-trees. For a given t,

$$g(i,0,t)=\min(g(i-1,0,t), g(i-1,0,t-1)+\alpha_i) \tag{18}$$

$$g(i,1,t)=\min(g(i-1,1,t), g(i-1,1,t-1)+\alpha_i+\beta_{1,i}) \tag{19}$$

The trees corresponding to $g(i,\cdot,\cdot)$ and $g(i-1,\cdot,\cdot)$ are shown in Figure 3(b) for $i=4$. When t nodes are to be chosen from $\{1,\ldots,i\}$ with $c_1=0$, c_i can take values either 0 or 1. If $c_i=0$, we must select t nodes from $\{1,\ldots,i-1\}$, while if $c_i=1$, we must select $t-1$ nodes from $\{1,\ldots,i-1\}$. If $c_i=0$, there is no additional cost incurred while when $c_i=1$, an additional cost of α_i is incurred. These two cases give rise to Equation (18). Note that the range of permissible value of t varies for these two cases and can be similar identified as in the recursions in the case of the series graph. For example, for the case $c_1=0$ and $c_i=0$, t can only range from 0 to $i-2$ in $g(i,0,t)$, while for $c_1=0$ and $c_i=1$, t can range from 1 to $i-1$. Using similar logic, Equation (19) can be written for the case where $c_1=1$. Finally, Q_{star} is obtained by looking at all possible values of $g(n,\cdot,\cdot)$ as follows:

$$Q_{star}=\min\left(\min_{t_1\in[k,n-1]} g(n,0,t_1), \min_{t_2\in[k,n]} g(n,1,t_2)\right) \tag{20}$$

This gives rise to the linear programming formulation by linearization as in Proposition 1. □

2.3. Formulation for General Trees

We now provide the solution to the problem on general trees. The series and star graphs may be viewed as two extreme cases of general trees, considering the height of the tree and number of children of the root node. The series graph has the maximum height with the root node containing exactly one child, while the star graph has the least height with the root node containing $n-1$ children. The recursions for the series and star graphs therefore help in the design of an algorithm

for general trees. The algorithm for a series graph involved a bottom-up traversal from the leaf node to the root while the star graph algorithm implicitly involved a traversal from the left most node to the right most node. The dynamic programming algorithm for general trees will involve solving sub-problems on a left to right as well as bottom up traversal of the nodes of the tree. Given a tree $T=(V,E)$, we are particularly interested in solving (7).

THEOREM 1. *The value of Q_T in (7) can be obtained using the following linear program:*

$$
\begin{aligned}
Q_T = &\max_{z,\mathbf{x}} z \\
&s.t\ z \le x_{1,d_1,0,t},\ \ t \in [k, n-1] \\
&\quad z \le x_{1,d_1,1,t},\ \ t \in [k, n] \\
&\quad x_{i,s,0,0} = 0 \text{ for } i \in V(i,s),\ s = [0, d_i] \\
&\quad x_{i,s,1,1} = \alpha_i \text{ for } i \in V(i,s),\ s = [0, d_i] \\
&\quad \textit{For each internal node } i: \\
&\quad\quad x_{i,1,0,t} \le x_{i(1),d_{i(1)},0,t} \text{ for } t = [0, N(i,1)-2] \qquad (21) \\
&\quad\quad x_{i,1,0,t} \le x_{i(1),d_{i(1)},1,t} \text{ for } t = [1, N(i,1)-1] \qquad (22) \\
&\quad\quad x_{i,1,1,t} \le x_{i(1),d_{i(1)},0,t-1} + \alpha_i \text{ for } t = [1, N(i,1)-1] \qquad (23) \\
&\quad\quad x_{i,1,1,t} \le x_{i(1),d_{i(1)},1,t-1} + \alpha_i + \beta_{i,i(1)} \text{ for } t = [2, N(i,1)] \qquad (24) \\
&\quad \textit{For each internal node } i \textit{ with out-degree at least 2:} \\
&\quad\quad x_{i,s,0,t} \le x_{i,s-1,0,t-a} + x_{i(s),d_{i(s)},0,a} \text{ for } t = [0, N(i,s)-2], a \in [a^1_{min}, a^1_{max}] \qquad (25) \\
&\quad\quad x_{i,s,0,t} \le x_{i,s-1,0,t-a} + x_{i(s),d_{i(s)},1,a},\ \text{for } t = [1, N(i,s)-1], a \in [a^2_{min}, a^2_{max}] \qquad (26) \\
&\quad\quad x_{i,s,1,t} \le x_{i,s-1,1,t-a} + x_{i(s),d_{i(s)},0,a},\ \text{for } t = [1, N(i,s)-1], a \in [a^3_{min}, a^3_{max}] \qquad (27) \\
&\quad\quad x_{i,s,1,t} \le x_{i,s-1,1,t-a} + x_{i(s),d_{i(s)},1,a} + \beta_{i,i(s)},\ \text{for } t = [2, N(i,s)], a \in [a^4_{min}, a^4_{max}] \qquad (28)
\end{aligned}
$$

where,

$a^1_{min} = \max(0, t-(N(i,s-1)-1)), a^1_{max} = \min(N(i(s), d_{i(s)})-1, t),$
$a^2_{min} = \max(1, t-(N(i,s-1)-1)), a^2_{max} = \min(N(i(s), d_{i(s)}), t),$
$a^3_{min} = \max(0, t-(N(i,s-1))), a^3_{max} = \min(N(i(s), d_{i(s)}-1), t-1),$
$a^4_{min} = \max(1, t-(N(i,s-1))), a^4_{max} = \min(N(i(s), d_{i(s)}), t-1).$

Proof: Assume the tree is endowed with a fixed ordering of the nodes. For any node i, we denote by $x_{i,s,y,t}$ the minimum value of the sub-problem where: (1) the vertices and edges are restricted to the tree $T(i,s)$ rooted at i and containing the first s sub-trees of i (as per the given ordering), (2) c_i takes a value $y \in \{0,1\}$ and (3) exactly t nodes in $T(i,s)$ take a value of 1. This is given as:

$$x_{i,s,y,t} = \min \left\{ \sum_{l \in V(i,s)} \alpha_l c_l + \sum_{(l,j) \in V(i,s)} \beta_{lj} c_l c_j : \textstyle\sum_{l \in V(i,s)} c_l = t, c_i = y, c_l \in \{0,1\} \forall l \in V(i,s) \right\}$$

where $s \in \{0, \ldots, d_i\}$, $y \in \{0,1\}$, $t \in \{y, \ldots, N(i,s) - 1 + y\}$. Then the optimal value of the overall problem (7) can be expressed in terms of the sub-problems as follows:

$$Q_T \leq x_{1,d_1,0,t},\ t \in [k, n-1]$$
$$Q_T \leq x_{1,d_1,1,t}\ t \in [k, n]$$

Note the similarity of the above updates with the updates for Q_{star} on star graphs. The function $g(\cdot,\cdot,\cdot)$ previously considered, looked at the sub-tree rooted at node 1 always. The first argument to $g(\cdot,\cdot,\cdot)$ considered $n-1$ values in $[2,n]$ corresponding to the children of node 1. The second index in $x_{1,\cdot,\cdot,\cdot}$ captures similar information.

We will now discuss the recursions to compute $\mathbf{x}$. The base cases include similar base cases as the star graph and the series case as follows:

$$x_{i,s,0,0} = 0,\ \ i \in V(i,s),\ s \in [0, d_i]$$
$$x_{i,s,1,1} = \alpha_i,\ \ i \in V(i,s),\ s \in [0, d_i]$$

The value $x_{i,s,0,0}$ looks at the case where $c_i = 0$ and the entire sub-tree rooted at i containing the first s subtrees of i have 0 nodes selected. This automatically suggests that for every j in $V(i,s)$, $c_j = 0$. In this case the optimal value of the sub-problem is 0. On the other hand, $x_{i,s,1,1}$ deals with the case where $c_i = 1$ and exactly one of the nodes in $T(i,s)$ is set to 1 (and this node has to be c_i by definition). Therefore the optimal value is α_i. This is true for every valid value of s.

Next, we will consider the recursions for $x_{i,1,\cdot,\cdot}$ which only looks at the tree rooted at i containing all nodes below and including the first child of i. The recursions are provided by Equations (21) to (24) and follow from the underlying recursions in the series graph. For an internal node i, let $j = i(1)$ denote its first child. Since $c_j \in \{0,1\}$, $x_{i,1,0,t}$ must take the minimum value out of $x_{j,d_j,0,t}$ and $x_{j,d_j,1,t}$ for all feasible values of t, while $x_{i,1,1,t}$ must take the minimum value from $x_{j,d_j,0,t-1}$ and $x_{j,d_j,1,t-1}$. These recursions are same as the recursions in Equations (11) and (12) for $\bar{f}(\cdot,\cdot,\cdot)$ with $i+1$ replaced with the first child $i(1)$.

Now for an internal node i with at least 2 children, we compute $x_{i,s,\cdot,\cdot}$ using $x_{j,\cdot,\cdot,\cdot}$ corresponding to all the children j of i, as well as $x_{i,s-1,\cdot,\cdot}$. The two sub-trees involved in computing $x_{i,s,\cdot,\cdot}$ are depicted as shaded regions (labelled T1 and T2) in Figure 4, for $s = 3$. First, suppose $c_i = 0$. We are interested to compute $x_{i,s,0,t}$. The t nodes to be selected from the tree rooted at node i can be split between T_1 and T_2 in various ways. The value of t itself can only range between 0 to $N(i,s) - 1$ since $c_i = 0$. Suppose a nodes are selected in T_2 and $t - a$ nodes are selected in T_1. The range of a differs based on whether the root node of T_2 is selected or not. Suppose the root node of T_2 is not selected (that is, $c_{i(s)} = 0$). Since the number of nodes selected from T_2 is a and $i(s)$ is not selected,

$$0 \leq a \leq N_{T_2} - 1$$

where N_{T_2} is the number of nodes in T_2. Also since the number of nodes selected from T_1 is $t-a$ and i is not selected,

$$0 \leq t-a \leq N_{T_1}-1$$

Based on the above two inequalities, we get $a \in [a^1_{min}, a^1_{max}]$, where $a^1_{min} = \max(0, t-(N_{T_1}-1))$ and $a^1_{max} = \min(N_{T_2}-1, t)$. For these values of a, the cost is just $x_{i,s-1,0,t-a} + x_{i(s),d_{i(s)},0,a}$ and no additional cost gets added as both i and $i(s)$ are not selected. Hence we get Equation (25). A similar treatment gives us Equation (26) corresponding to $c_{i(s)} = 1$. Note that the cost $\alpha_{i(s)}$ is already part of $x_{i(s),d_{i(s)},0,a}$ and does not need to be explicitly added. $x_{i,s,0,t}$ must take the minimum value of all terms in the RHS of Equations (25) and (26).

Now suppose $c_i = 1$, we are interested to compute $x_{i,s,1,t}$. The valid values of t range from 1 to $N(i,s)$. Here we will illustrate the case where $c_{i(s)} = 1$. Again assume the sub-tree T_2 rooted at $i(s)$ contains a nodes and T_1 contains $t-a$ nodes. Using similar reasoning from the earlier step, the range of valid values for a can be derived. Now in addition to the sub-tree costs $x_{i,s-1,1,t-a} + x_{i(s),d_{i(s)},1,a}$, we also incur an additional cost $\beta_{i,i(s)}$ of selecting both nodes i as well as $i(s)$. Hence we get Equation (28). The case where $c_{i(s)} = 0$ follows using similar logic(see (27)). Note that the individual item costs α_i and $\alpha_{i(s)}$ are absorbed in $x_{i,s-1,1,t-a}$ and $x_{i(s),d_{i(s)},1,a}$ respectively. Finally, $x_{i,s,1,t}$ must take the minimum value of all terms in the RHS of Equations (27) and (28) and hence the inequalities arise. The variable z denotes Q_T. □

The particular sub-trees used in these cases illustrated in Figure 4 can be viewed as generalizations of the sub-trees used in the star graph. For example, in Figure 3(b), the darker sub-tree (with nodes $\{1,2,3\}$) represents $T1$ and the sub-tree T_2 (not shown explicitly) trivially has exactly 1 node (node 4 in the particular instance in the Figure 3(b)).

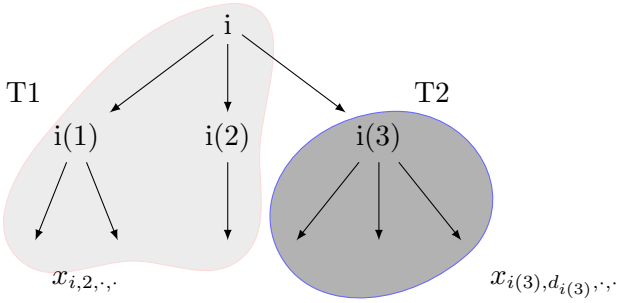


Figure 4 $x_{i,3,\cdot,t}$ **can be computed using** $x_{i,2,\cdot,t-a}$ **(corresponding to the sub-tree T1) and** $x_{i(3),d_{i(3)},\cdot,a}$ **(corresponding to the sub-tree T2) so that** $t-a$ **nodes are selected from T1 and** a **nodes are selected from T2.**

The number of variables in the optimization is $\sum_{i=1}^{n} 2d_i n$ or $O(n^2)$ (as $\sum_i d_i = O(n)$ for a tree) while the number of constraints is $O(n^3)$.

3. Probability Bounds with Trees

We will now use the results from the previous section to derive tight bounds for the probability $U(k)$ when the univariate and bivariate probabilities for a given tree graph $T=(V,E)$ are known. The exponential sized dual linear program involves two sets of constraints (5) and (6) to compute $U(k)$. The results in Section 2 discuss the separation problem involving (5). We will now see that (6) can be equivalently represented by a set of linear constraints.

LEMMA 1. *For a given set of values for $\{\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$, constraint (6) is equivalent to verifying the feasibility of the following set of linear constraints in the variables $\{\boldsymbol{\eta}, \boldsymbol{\gamma}, \boldsymbol{\Delta}, \boldsymbol{\tau}, \boldsymbol{\chi}\}$:*

$$\lambda - \sum_{(i,j)\in E} (\Delta_{ij} + \chi_{ij}) - \sum_{i=1}^{n} \tau_i \geq 0 \tag{29}$$

$$\sum_{j:(i,j)\in E} (\Delta_{ij} - \eta_{ij}) + \sum_{j:(j,i)\in E} (\Delta_{ji} - \gamma_{ji}) + \tau_i + \alpha_i \geq 0 \quad \textit{for } i \in [n] \tag{30}$$

$$\eta_{ij} + \gamma_{ij} - \Delta_{ij} + \chi_{ij} + \beta_{ij} \geq 0 \quad \textit{for } (i,j) \in E \tag{31}$$

$$\boldsymbol{\eta}, \boldsymbol{\gamma}, \boldsymbol{\Delta}, \boldsymbol{\tau}, \boldsymbol{\chi} \geq 0 \tag{32}$$

Proof: Constraint (6) may be re-written as,

$$\lambda - \max_{\mathbf{c}\in\{0,1\}^n} \left\{ -\textstyle\sum_{i=1}^{n} \alpha_i c_i - \sum_{(i,j)\in E} \beta_{ij} c_i c_j : c_i \in \{0,1\} \forall\, i \right\} \geq 0 \tag{33}$$

Note that for a given set of values for $\{\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$, this reduces to optimizing a quadratic function over the extreme points of the unit hypercube which corresponds to optimization over the Boolean quadric polytope. We can now directly apply the results in Padberg [1989] who derived a tight formulation when the graph is a tree. The results therein implies the following LP relaxation is tight for the maximization problem in Equation (33) when the sparsity pattern of the quadratic terms $\boldsymbol{\beta}$ is given as tree $T=(V,E)$:

$$\begin{array}{ll} \max\limits_{\mathbf{c},\mathbf{y}} & -\sum_{i=1}^{n} \alpha_i c_i - \sum_{(i,j)\in T} \beta_{ij} y_{ij} \\ \text{s.t} & y_{ij} - c_i \leq 0 \ \text{ for } (i,j) \in E \\ & y_{ij} - c_j \leq 0 \ \text{ for } (i,j) \in E \\ & c_i + c_j - y_{ij} \leq 1 \ \text{ for } (i,j) \in E \\ & 0 \leq c_i \leq 1 \ \text{ for } i \in [n] \\ & 0 \leq y_{ij} \leq 1 \ \text{ for } (i,j) \in E \end{array}$$

The constraints (30), (31) and (32) in Lemma 1 correspond to the constraints in the dual of the above linear optimization problem. Constraint (29) appears as a result of forcing the dual objective to be non-negative (due to the non-negativity requirement of the objective in (33)). □We are now ready to provide the main result of the paper.

THEOREM 2. *Consider an n-dimensional Bernoulli random vector $\tilde{\mathbf{c}}$ where the univariate probabilities $p_i = P(\tilde{c}_i = 1)$ for $i = 1, \ldots, n$ and the bivariate marginals $p_{ij} = P(\tilde{c}_i = 1, \tilde{c}_j = 1)$ for all*

$(i,j) \in E$ for a tree graph are specified. Let U_k^* denote the optimal value of the following linear program over the decision variables $\Gamma = \{\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Delta}, \boldsymbol{\eta}, \boldsymbol{\chi}, \boldsymbol{\tau}, \boldsymbol{x}, z\}$:

$$
\begin{aligned}
U_k^* = \min_{\Gamma}\ & \lambda + \sum_{i=1}^{n} \alpha_i p_i + \sum_{(i,j)\in E} \beta_{ij} p_{ij} \\
s.t.\ & \lambda - \sum_{(i,j)\in E} (\Delta_{ij} + \chi_{ij}) - \sum_{i=1}^{n} \tau_i \geq 0 \\
& \sum_{j:(i,j)\in E} (\Delta_{ij} - \eta_{ij}) + \sum_{j:(j,i)\in E} (\Delta_{ji} - \gamma_{ji}) + \tau_i + \alpha_i \geq 0 \ \textit{for } i \in [n] \\
& \eta_{ij} + \gamma_{ij} - \Delta_{ij} + \chi_{ij} + \beta_{ij} \geq 0 \ \textit{for } (i,j) \in E \\
& \lambda + z \geq 1 \\
& x_{1,d_1,0,t} - z \geq 0 \ \textit{for } t \in [k,n] \\
& x_{1,d_1,1,t} - z \geq 0 \ \textit{for } t \in [k,n] \\
& x_{i,s,0,0} = 0 \ \ \textit{for } i \in [n], \textit{ for } s \in [0,d_i] \\
& x_{i,s,1,1} - \alpha_i = 0 \ \textit{for } i \in [n], \textit{ for } s \in [0,d_i] \\
& \textit{For each internal node } i\textit{:} \\
& \quad x_{i(1),d_{i(1)},0,t} - x_{i,1,0,t} \geq 0 \ \textit{for } t \in [0, N(i,1)-2] \\
& \quad x_{i(1),d_{i(1)},1,t} - x_{i,1,0,t} \geq 0 \ \textit{for } t \in [1, N(i,1)-1] \\
& \quad x_{i(1),d_{i(1)},0,t-1} - x_{i,1,1,t} + \alpha_i \geq 0 \ \textit{for } t \in [1, N(i,1)-1] \\
& \quad x_{i(1),d_{i(1)},1,t-1} - x_{i,1,1,t} + \alpha_i + \beta_{i,i(i)} \geq 0 \ \textit{for } t \in [2, N(i,1)] \\
& \textit{For each internal node } i \textit{ with out-degree at least 2:} \\
& \quad x_{i,s-1,0,t-a} + x_{i(s),d_{i(s)},0,a} - x_{i,s,0,t} \geq 0 \ \textit{for } s = [2,d_i], t = [0,(N(i,s)-2)], a = [a^1_{min}, a^1_{max}] \\
& \quad x_{i,s-1,0,t-a} + x_{i(s),d_{i(s)},1,a} - x_{i,s,0,t} \geq 0 \ \textit{for } s = [2,d_i], t = [1,(N(i,s)-1)], a = [a^2_{min}, a^2_{max}] \\
& \quad x_{i,s-1,1,t-a} + x_{i(s),d_{i(s)},0,a} - x_{i,s,1,t} \geq 0 \ \textit{for } s = [2,d_i], t = [1,(N(i,s)-1)], a = [a^3_{min}, a^3_{max}] \\
& \quad x_{i,s-1,1,t-a} + x_{i(s),d_{i(s)},1,a} - x_{i,s,1,t} + \beta_{i,i(s)} \geq 0 \ \textit{for } s = [2,d_i], t = [2,(N(i,s))], a = [a^4_{min}, a^4_{max}] \\
& \boldsymbol{\eta}, \boldsymbol{\gamma}, \boldsymbol{\Delta}, \boldsymbol{\tau}, \boldsymbol{\chi} \geq 0
\end{aligned}
$$

where $a^1_{min} = \max(0, t - (N(i,s-1)-1)), a^1_{max} = \min(N(i(s), d_{i(s)} - 1), t)$,

$a^2_{min} = \max(1, t - (N(i,s-1)-1)), a^2_{max} = \min(N(i(s), d_{i(s)}, t)$,

$a^3_{min} = \max(0, t - (N(i,s-1))), a^3_{max} = \min(N(i(s), d_{i(s)} - 1), t-1)$,

$a^4_{min} = \max(1, t - (N(i,s-1))), a^4_{max} = \min(N(i(s), d_{i(s)}), t-1)$.

Then, $U(k) = U_k^*$.

Proof: We derive the linear programming reformulation by considering each of the two groups of constraints (5) and (6) that arise in the dual problem. Constraint (5) can be re-written as,

$$\lambda + \min_{\mathbf{c}\in\{0,1\}^n} \left\{ \textstyle\sum_{i=1}^n \alpha_i c_i + \sum_{(i,j)\in T} \beta_{ij} c_i c_j : \sum_{i=1}^n c_i \geq k \right\} \geq 1$$

This is the cardinality constrained quadratic knapsack problem in minimization form. In Theorem 1, we provided a linear programming reformulation for this sub-problem. Plugging in the constraints from the linear program in Theorem 1 and forcing the objective value to be greater than 1 gives us all, except the first three constraints in the linear program. The first three constraints are equivalent to Equation (6) as stated in Lemma 1. This completes the proof of Theorem 2. □

The number of variables and constraints in our linear programming solution are $O(n^2)$ and $O(n^3)$ respectively, which implies that the tight bound is solvable in polynomial time.

3.1. Lower Bound

The analogous approach can be used to find lower bounds $L(k)$ for the same sum on the given tree graph $T=(V,E)$. To see this, let $\tilde{d}_i = 1-\tilde{c}_i$ and define $q_i = 1-p_i$ and $q_{ij} = \mathbb{P}(\tilde{c}_i = 0, \tilde{c}_j = 0)$ for all $(i,j) \in E$. Then,

$$\begin{aligned} L(k) = \min \mathbb{P}_\theta(\textstyle\sum_{i=1}^n \tilde{c}_i \geq k) &= 1 + \min\left(-\mathbb{P}_\theta(\textstyle\sum_{i=1}^n \tilde{c}_i < k)\right) \\ &= 1 - \max \mathbb{P}_\theta(\textstyle\sum_{i=1}^n 1-\tilde{c}_i \geq n-k+1) \\ &= 1 - \max \mathbb{P}_\theta(\textstyle\sum_{i=1}^n \tilde{d}_i \geq n-k+1) = U(n-k+1) \end{aligned} \tag{34}$$

where $U(n-k+1)$ must be computed for the given tree graph T by setting $p_i = q_i$ and $p_{ij} = q_{ij}$ in Theorem 2.

3.2. Probability with a Tree Graphical Model

We now relate our bounds to the computation of $\mathbb{P}(\sum_{i=1}^n \tilde{c}_i \geq k)$ under the same information as before but now focus on a tree graphical model where conditional independence is assumed. The difference from the bounds in the paper is that this induces a unique distribution where every random variable $\tilde{c}_i$ is independent of all its siblings, conditional on knowledge of the realization of its parent i. The next proposition provides a dynamic programming recursion (similar to Section 2) to compute $\mathbb{P}(\sum_{i=1}^n \tilde{c}_i \geq k)$ in this case.

PROPOSITION 3. *Consider a n dimensional Bernoulli random vector $\tilde{\mathbf{c}}$ where the univariate probabilities $p_i = P(\tilde{c}_i = 1)$ for $i = 1,\ldots,n$ and the bivariate marginals $p_{ij} = P(\tilde{c}_i = 1, \tilde{c}_j = 1)$ for all $(i,j) \in E$ for a tree graph are specified and assume it is a conditionally independent distribution on the tree as in (1). Then, the following recursions can be used to compute $\mathbb{P}(\sum_{i=1}^n \tilde{c}_i \geq k)$:*

$$w_{i,s,0,0} = p_0(i) \quad \textit{for } i \in [n], s \in [0, d_i] \tag{35}$$

$$w_{i,s,1,1} = p_1(i) \quad \textit{for } i \in [n], s \in [0, d_i] \tag{36}$$

For each internal node i,

$$\begin{aligned} w_{i,1,0,t} = &\frac{w_{i(1),d_{i(1)},0,t} \times p^{00}(i,i(1)) \times \mathbb{1}\{t \in [0, N(i,1)-1]\}}{p_0(i(1))} \\ &+ \frac{w_{i(1),d_{i(1)},1,t} \times p^{01}(i,i(1)) \times \mathbb{1}\{t \in [1, N(i,1)-1]\}}{p_1(i(1))}, \ t \in [0, N(i,1)-1] \end{aligned} \tag{37}$$

$$\begin{aligned} w_{i,1,1,t} = &\frac{w_{i(1),d_{i(1)},0,t-1} \times p^{10}(i,i(1)) \times \mathbb{1}\{t \in [1, N(i,1)-1]\}}{p_0(i(1))} \\ &+ \frac{w_{i(1),d_{i(1)},1,t-1} \times p^{11}(i,i(1)) \times \mathbb{1}\{t \in [2, N(i,1)]\}}{p_1(i(1))}, \ t \in [1, N(i,1)-1] \end{aligned} \tag{38}$$

For each internal node i with out-degree at least 2:

$$\begin{aligned} w_{i,s,0,t} = &\sum_{a=a^1_{min}}^{a^1_{max}} \frac{w_{i,s-1,0,t-a} \times w_{i(s),d_{i(s)},0,a} \times p^{00}(i,i(s)) \mathbb{1}\{t \in [0, (N(i,s)-2)]\}}{p_0(i) p_0(i(s))} \\ &+ \sum_{a=a^2_{min}}^{a^2_{max}} \frac{w_{i,s-1,0,t-a} \times w_{i(s),d_{i(s)},1,a} \times p^{01}(i,i(s)) \mathbb{1}\{t \in [1, (N(i,s)-1)]\}}{p_0(i) p_1(i(s))} \end{aligned}$$

$$\text{for } s \in [2, d_i], t \in [0, N(i,s)-1] \tag{39}$$

$$w_{i,s,1,t} = \sum_{a=a^3_{min}}^{a^3_{max}} \frac{w_{i,s-1,1,t-a} \times w_{i(s),d_{i(s)},0,a} \times p^{10}(i,i(s)) \mathbb{1}\{t \in [1,(N(i,s)-1)]\}}{p_1(i)p_0(i(s))} + \sum_{a=a^4_{min}}^{a^4_{max}} \frac{w_{i,s-1,1,t-a} \times w_{i(s),d_{i(s)},1,a} \times p^{11}(i,i(s)) \mathbb{1}\{t \in [2,N(i,s)]\}}{p_1(i)p_1(i(s))}$$

$$\text{for } s \in [2, d_i], t \in [1, N(i,s)] \tag{40}$$

where $a^1_{min} = \max(0, t-(N(i,s-1)-1)), a^1_{max} = \min(N(i(s), d_{i(s)}-1), t)$,
$a^2_{min} = \max(1, t-(N(i,s-1)-1)), a^2_{max} = \min(N(i(s), d_{i(s)}), t)$,
$a^3_{min} = \max(0, t-(N(i,s-1))), a^3_{max} = \min(N(i(s), d_{i(s)}-1), t-1)$,
$a^4_{min} = \max(1, t-(N(i,s-1))), a^4_{max} = \min(N(i(s), d_{i(s)}), t-1)$,
$p^{00}(i,j) = \mathbb{P}(\tilde{c}_i = 0, \tilde{c}_j = 0) = 1 - p_i - p_j + p_{ij}$,
$p^{01}(i,j) = \mathbb{P}(\tilde{c}_i = 0, \tilde{c}_j = 1) = p_j - p_{ij}$,
$p^{10}(i,j) = \mathbb{P}(\tilde{c}_i = 1, \tilde{c}_j = 0) = p_i - p_{ij}$, $p^{11}(i,j) = p_{ij}$ *and* $p_1(i) = p_i, p_0(i) = 1 - p_i$.

In particular, $\mathbb{P}(\sum_{i=1}^n \tilde{c}_i \geq k) = \sum_{t=k}^n w_{1,d_1,0,t} + w_{1,d_1,1,t}$.

The proof idea is to express the probability $\mathbb{P}\left(\sum_{l \in T(i,s)} \tilde{c}_l = t, \tilde{c}_i = y\right)$, denoted by $w_{i,s,y,t}$, in terms of probabilities for smaller sub-trees of $T(i,s)$. The details are provided in the online companion.

The approach for deriving the recursions on $\mathbf{w}$ progresses in a very similar manner to the recursions for $\mathbf{x}$ in the quadratic knapsack problem on tree graphs in Section 2.3. Both approaches work on $O(n^2)$ variables. The same sub-trees (as depicted in Figure 4) are used in the computation of $x_{i,s,y,t}$ and $w_{i,s,y,t}$ in Theorems 1 and 3 respectively. For example, in (40), the computation of $w_{i,s,1,t}$ makes use of sums and products involving $w_{i,s-1,0,t-a}, w_{i(s),d_{i(s)},0,a}$ and $w_{i(s),d_{i(s)},1,a}$. In a similar vein, the computation of $x_{i,s,1,t}$ in Theorem 2 makes use of inequalities and sums involving $x_{i,s-1,0,t-a}, x_{i(s),d_{i(s)},0,a}$ and $x_{i(s),d_{i(s)},1,a}$. For the sub-problems involving only the first sub-tree in (37), $w_{i,1,0,t}$ makes use of summations involving $w_{i(1),d_{i(1)},0,t}$ and $w_{i(1),d_{i(1)},1,t}$. Similarly the computation of $x_{i,1,0,t}$ makes use of inequalities involving $x_{i(1),d_{i(1)},0,t}$ and $x_{i(1),d_{i(1)},1,t}$. Of course, the expressions themselves are different in the two theorems - Theorem 1 looks at solving an underlying optimization problem while the goal of Proposition 3 is to simplify the computation of a probability which otherwise involves an exponential number of operations. A comparison of the underlying techniques proposed for these settings is provided in Table 2.

4. Generalization to Bounds for Weighted Sum of Probabilities

Our approach can be generalized to compute upper and lower bounds for the weighted sum of probabilities $\sum_{s=0}^n w_s \mathbb{P}(\sum_{i=1}^n \tilde{c}_i = s)$ with a given weight vector $\mathbf{w} \in \mathbb{R}^+_{n+1}$. Weighted sums arise in a scenario where, for example, the set $\{1, \ldots, n\}$ can be partitioned into two disjoint sets A

Table 2 **Comparison of proposed approach for computing the upper bound** $\max \mathbb{P}(\sum_{i=1}^{k} \tilde{c}_i \geq k)$ **vs** $\mathbb{P}(\sum_{i=1}^{k} \tilde{c}_i \geq k)$ **for the conditionally independent distribution on a given tree**

	Upper Bound	**Conditional Independence**
Formulation	Linear program	Dynamic programming recursion
No. of variables	$O(n^2)$	$O(n^2)$
No. of operations/constraints	$O(n^3)$ constraints in the formulation	$O(n^3)$ summations in Equations (37) to (40)
Type of operations	Inequalities involving summations over the variables	Equalities involving summations and products over the variables

and B such that the random variables $\{\tilde{c}_i, i \in A\}$ are known to be mutually independent and also independent from $\{\tilde{c}_j, j \in B\}$. The random variables corresponding to the set B could however be dependent but the joint distribution over $\{\tilde{c}_j, j \in B\}$ is unknown. Denote by $\tilde{\mathbf{c}}_A$ the vector of random variables corresponding to the set A. In particular, assume the following information:

1. A tree structure $T_B = (B, E_B)$
2. The univariate probabilities $\mathbb{P}(\tilde{c}_i = 1) = p_i$, for $i \in [n]$
3. The bivariate probabilities $\mathbb{P}(\tilde{c}_i = 1, \tilde{c}_j = 1) = p_{ij}$ for $(i,j) \in E_B$
4. $\mathbb{P}(\tilde{c}_i = 1, \tilde{c}_j = 1) = p_i p_j$ for $i \in A, j \in B$
5. $\mathbb{P}(\tilde{\mathbf{c}}_A = \mathbf{r}_A) = \prod_{i \in A} \mathbb{P}(\tilde{c}_i = r_i)$ for $\mathbf{r}_A \in \{0,1\}^{|A|}$

By enumerating all ways in which $\sum_{i \in A} \tilde{c}_i$ and $\sum_{i \in B} \tilde{c}_i$ add up to a value $j \geq k$ we can express the tail probability as a weighted sum as below,

$$\mathbb{P}(\sum_{i=1}^{n} \tilde{c}_i \geq k) = \sum_{s=0}^{|B|} w_s \mathbb{P}(\sum_{i \in B} \tilde{c}_i = s)$$

where $w_s = \mathbb{P}(\sum_{i \in A} \tilde{c}_i \geq k - s) \geq 0$. This relation follows as a consequence of independence between $\tilde{\mathbf{c}}_A$ and $\tilde{\mathbf{c}}_B$. $\sum_{i \in A} \tilde{c}_i$ is a sum of independent but non-identical Bernoulli random variables and therefore takes a Poisson-binomial distribution, for which the probability w_s can be computed in a recursive manner in polynomial time (see Chen [1998]).

We now show that the computation of $\max_{\theta \in \Theta} \sum_{s=0}^{n} w_s \mathbb{P}(\sum_{i=1}^{n} \tilde{c}_i = s)$, given any $\mathbf{w} \in \mathbb{R}^{+}_{n+1}$ can be easily done. The result in Theorem 2 provides a linear program for the special case where $w_s = 0$ for $s \in [0, k-1]$ and $w_s = 1$ for $s \in [k, n]$.

Given any $\mathbf{w} \in \mathbb{R}^{+}_{n+1}$, the analogous exponential sized dual formulation is:

$$\min_{\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta}} \lambda + \sum_{i=1}^{n} \alpha_i p_i + \sum_{(i,j) \in E} \beta_{ij} p_{ij}$$

$$\text{s.t.} \lambda + \sum_{i=1}^{n} \alpha_i c_i + \sum_{(i,j) \in E} \beta_{ij} c_i c_j \geq w_s, \quad \text{for } \mathbf{c} \in \{0,1\}^n \text{ where} \sum_{i=1}^{n} c_i = s, \text{for } s \in [0,n] \tag{41}$$

$$\lambda + \sum_{i=1}^{n} \alpha_i c_i + \sum_{(i,j) \in E} \beta_{ij} c_i c_j \geq 0 \quad \text{for } \mathbf{c} \in \{0,1\}^n \tag{42}$$

Constraint (42) is exactly the same as constraint (6) and therefore Lemma 1 gives a reformulation for this constraint. For constraint (41), for each value of s, a polynomial sized linear programming

formulation can be derived based on similar dynamic programming recursions in Section 2. The objective of the resulting linear program must be forced to take a value greater than w_s (instead of a value of 1 in Theorem 2).

5. Numerical Computations

We now present the results of our numerical computations. For convenience, in this section, we will use the notation $S = \sum_{i=1}^{n} \tilde{c}_i$. All computations were carried out using Gurobi version 9.0 [1] on a Python interface.

5.1. Bounds for various bivariate dependencies

In the first set of experiments, for $n = 15$ Bernoulli random variables, we considered 50 randomly generated trees with the univariate probabilities p_i generated uniformly in $(0, 0.1]$. We computed the following bounds for various values of k. (1) $U_{uv}(k)$ = Maximum value of $\mathbb{P}(\sum_{i=1}^{n} \tilde{c}_i \geq k)$ assuming univariate information alone (using (4)) We study the scenarios when the bivariate probabilities are generated using the comonotone and anti-comonotone copulas with the generated univariate probability distributions. The comonotone copula represents maximum positively dependent random variables while the anti-comonotone copula represents maximum negatively dependent random variables [Nelsen, 2006, Puccetti and Wang, 2015]. For Bernoulli random variables, it is known that for the comonotone copula, $p_{ij} = \min(p_i, p_j)$ while for the anti-comonotone copula, $p_{ij} = \max(p_i + p_j - 1, 0)$.

The range of values of $U(.)$, $U_{uv}(.)$ and P_{ci} over 50 runs are provided in Figure 5 (with labels Tree, Univar and Cond-ind, and shown using circles, squares and diamond markers respectively).

Our key observations, based on Figure 5 are as follows:

1. When the bivariate distributions are specified using the comonotone copula, the univariate bound is much larger than the tree bound for lower values of k whereas for larger values of k they almost co-incide. Thus the maximum utility of including bivariate information over univariate information is seen for lower values of k here. An intuitive explanation is the following: As the comonotone copula assigns larger probabilities for more random variables taking similar values, when no constraints on bivariates are specified, we expect the largest value of $\mathbb{P}(\sum_{i=1}^{n} \tilde{c}_i \geq k)$ to be attained by the comonotone copula for larger values of k. Therefore the worst case distribution for the univariate bound is consistent with the bivariate distributions assumed for the tree bounds.

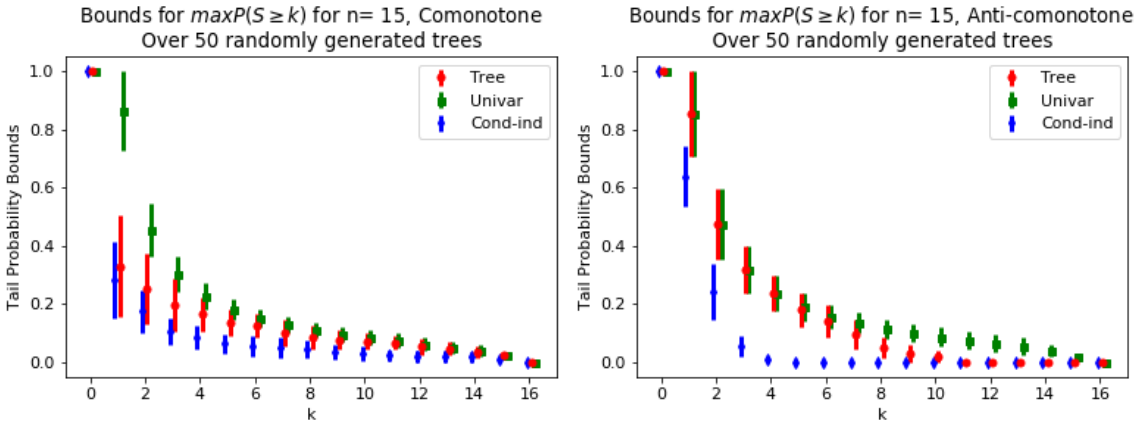
2. For the anti-comonotone copula, the tree bound is almost identical to the univariate bound for lower values of k and the values start differing as k becomes large. Therefore the maximum utility of including bivariate information over univariate information is seen for larger values of k here.

[1] www.gurobi.com

Thus the distribution that achieves the optimal univariate bound drifts closer to anti-comonotone distributions for smaller values of k.

3. The conditionally independent distribution gives a different bound from $U(.)$ and $U_{uv}(.)$. As it is a feasible distribution in Θ, it gives a lower bound to the optimal tree and univariate bounds.

Figure 5 Demonstration of the bounds for various dependence structures. The range of bounds U, U_{uv}, P_{ci} and U_{mc} over 50 arbitrary tree structures on $n = 15$ random variables is shown.



5.2. Computational Times

We now investigate the execution times of our compact linear program in Theorem 2. We compare the running times of our method against a row generation scheme to solve the dual problem in Section 1.3. In the row generation scheme, the reduced problem (43) is solved first. This reduced problem corresponds to the dual problem in Section 1.3 but only with reformulations for (6) included. The reduced set of $2n$ constraints ensure that all the variables are bounded. The remaining constraints necessary to enforce (5) are iteratively generated using Algorithm 1.

$$\begin{aligned}
&\min\ \lambda + \textstyle\sum_{i=1}^{n} \alpha_i p_i + \sum_{(i,j)\in E} \beta_{ij} p_{ij} \\
&\text{s.t. } \lambda - \sum_{(i,j)\in E} (\Delta_{ij} + \chi_{ij}) - \sum_{i=1}^{n} \tau_i \geq 0 \\
&\quad \sum_{j:(i,j)\in E} (\Delta_{ij} - \eta_{ij}) + \sum_{j:(j,i)\in E} (\Delta_{ji} - \gamma_{ji}) + \tau_i + \alpha_i \geq 0 \text{ for } i \in [n] \\
&\quad \eta_{ij} + \gamma_{ij} - \Delta_{ij} + \chi_{ij} + \beta_{ij} \geq 0 \text{ for } (i,j) \in E \\
&\quad \boldsymbol{\Delta}, \boldsymbol{\chi}, \boldsymbol{\tau}, \boldsymbol{\eta}, \boldsymbol{\gamma} \geq 0, \\
&\quad \lambda, \boldsymbol{\alpha}, \boldsymbol{\beta} \text{ unrestricted}
\end{aligned} \tag{43}$$

In Step 4 of Algorithm 1, a binary QP sub-problem is solved in order to identify a violated constraint. If no violated constraint is found, this shows that the original problem has been solved to optimality and the algorithm terminates returning the bound computed, otherwise an inequality is included to ensure the constraint is not violated and re-optimization is performed. Algorithm 1 provides the optimal probability provided the solution returned in Step 4 is correct. We explored the use of CPLEX v12.10 [2] as well as Gurobi solvers for the binary QP. We experimented with several

[2] www.cplex.com

Algorithm 1 Row generation scheme to compute $U(k)$

1: Let U^{sub} denote the optimization problem (43)
2: **loop**
3: Solve U^{sub} and let the solution obtained be $\lambda^*, \alpha^*, \beta^*, \Delta^*, \chi^*, \tau^*, \eta^*, \gamma^*$
4: Solve $\mathbf{c}^* = \arg\min \left\{ \lambda^* + \sum_{i=1}^n \alpha_i^* c_i + \sum_{(i,j)\in E} \beta_{ij}^* c_i c_j : \sum_i c_i \geq k, \mathbf{c} \in \{0,1\}^n \right\}$ ▷ *Binary QP*
5: **if** $\lambda^* + \sum_{i=1}^n \alpha_i^* c_i^* + \sum_{(i,j)\in E} \beta_{ij}^* c_i^* c_j^* \geq 1$ **then return** $\lambda^* + \sum_{i=1}^n \alpha_i^* p_i + \sum_{(i,j)\in E} \beta_{ij}^* p_{ij}$
6: **else**
7: Modify U^{sub} to include an extra constraint $\lambda + \sum_{i=1}^n \alpha_i c_i^* + \sum_{(i,j)\in E} \beta_{ij} c_i^* c_j^* \geq 1$
8: **end if**
9: **end loop**

parameters in CPLEX for the QP sub-problem in Step 4 (linearize switch to force linearization of quadratic terms, various levels of aggressiveness of BQP cuts [Bonami et al., 2018a] and RLT cuts, qpmakepsd options for convexifying the objective). For both CPLEX and Gurobi, we additionally explored the use of dual simplex method towards a faster re-optimization. However we found the performance with the default parameter settings the best in both solvers. To the best of our knowledge, in the default setting, CPLEX uses a classifier to decide whether the quadratic terms must be linearized or not [Bonami et al., 2018b]. If not linearization, the branch and bound approach is used. Gurobi uses the non-convex QP solver, by default. Between the default settings on both solvers, we did not find a significant difference in terms of performance. The QP solvers were found to be very efficient (An average time of about $0.001s$ was incurred for solving (7), for $n = 50$, over 100 randomly generated trees and $\boldsymbol{\alpha}, \boldsymbol{\beta}$, for each $k \in \{5, 20, 40, 45\}$).

5.2.1. Execution times as a function of the number of nodes: We first report the execution times, the number of constraints and variables in our compact LP as well as row generation as a function of the number of nodes for the series graph over 50 randomly generated instances of univariate probabilities. The comonotone distribution is used for the bivariate probabilities. Figure 6 shows the number of constraints (in the table) and the execution times for the series graph with k fixed to be 0.4n, while Figure 7 demonstrates the same for k fixed to a larger value of $0.8n$. Our findings are provided below:

1. Size and speed of the formulations: Table 3 provides the number of variables in both formulations for the series graph.The same numbers were obtained for all values of k, and were constants for a fixed n and a given tree structure. Row generation uses exactly $2n$ variables while compact LP, in general, uses $O(n^2)$ variables. From the tables along side Figures 6 and 7, we see that row generation finally generates much fewer constraints than our compact LP. For example from Figure 6 for $n = 50$, $k = int(0.4n)$, row generation produces an average and max of 869 and 1216

constraints while compact LP has 5260 constraints. However in spite of the final smaller formulation, we observe in the figure that the process of performing an iterative constraint selection leads to a much larger execution time for row generation.

2. Analysis for other tree structures: Tables 5 and 6 illustrate the execution time comparisons for random instances on balanced binary trees and star graphs respectively. The balanced binary tree has a lesser height than the series graph and the star graph has the least height. We observe that row generation can be extremely expensive in these scenarios too while our compact LP offers a significant computational advantage. For example,

(a) For $n = 90$ on balanced binary trees (Table 5), the largest time taken by compact LP was 3.8s while row generation required 564.4s (compact LP was about 148 times faster).

(b) For $n = 35$ on star trees (Table 6), the largest execution time of compact LP was 0.39s while for row generation, the largest execution time was 724.5s.

3. Scalability of compact LP: We also report the execution times for our compact LP separately in Table 4. To the extent possible, the experiments were performed over 50 randomly generated trees. However for $n = 200$ and $n = 300$, due to larger execution times, we have reported the results over 10 randomly generated trees. All execution times are reported in seconds unless explicitly stated. For even about $n = 150$ random variables, we find that our approach is quite fast and takes just about a minute on an average.

n	Compact LP	Row Generation		
		Mean	Min	Max
10	252	95	69	154
20	904	229	167	381
30	1956	411	275	634
40	3408	643	427	1012
50	5260	869	646	1216
60	7512	1177	795	1998
70	10164	1587	958	2769
80	13216	2043	1228	3694
90	16668	2532	1707	4769
100	20520	2991	1922	6052

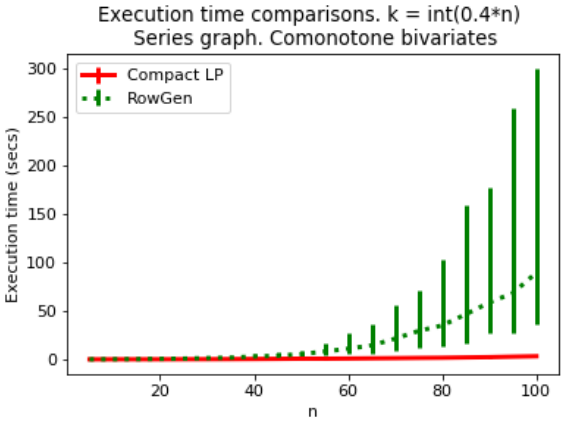


Figure 6 The table provides the number of constraints in the compact LP and the final linear program by row generation for the series tree, comonotone bivariates and $k = \mathbf{int}(0.4n)$ over 50 random instances while the figure illustrates the execution times of both approaches.

n	Compact LP	Row Generation Mean	Min	Max
10	244	52	45	62
20	888	136	108	172
30	1932	241	191	345
40	3376	365	301	478
50	5220	516	394	771
60	7464	669	494	899
70	10108	876	656	1351
80	13152	1155	849	3295
90	16596	1368	932	2145
100	20440	1546	1197	2327

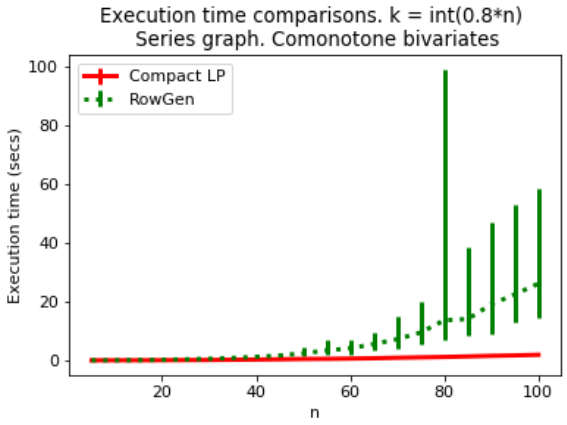


Figure 7 **The table provides the number of constraints in the compact LP and the final linear program by row generation for the series tree, comonotone bivariates and $k = \mathbf{int}(0.8n)$ over 50 random instances while the figure illustrates the execution times of both approaches.**

n	Compact LP	Row Generation
10	256	20
20	716	40
30	1376	60
40	2236	80
50	3296	100
60	4556	120
70	6016	140
80	7676	160
90	9536	180
100	11596	200

Table 3: Number of variables in the compact linear program vs row generation, for series graph.

n	Mean	Min	Max	No. of instances
50	0.571	0.380	0.719	50
80	2.892	1.774	5.000	50
100	6.582	3.618	19.062	50
120	17.138	9.395	34.844	50
150	59.813	22.552	244.952	50
200	8.195 mins	1.79 mins	19.609 mins	10
300	66.916 mins	8.338 mins	2.817 hrs	10

Table 4: Execution times of our approach for various n over randomly generated trees and comonotone bivariates. The value of k used here is $k = int(0.4n)$. All measurements are in seconds unless explicitly stated.

	Compact LP			Row Generation		
n	Mean	Min	Max	Mean	Min	Max
10	0.016	0.013	0.047	0.105	0.035	0.166
20	0.062	0.052	0.094	0.632	0.193	1.413
30	0.139	0.116	0.223	1.872	0.867	4.429
40	0.293	0.245	0.455	5.146	2.402	17.214
50	0.458	0.398	0.671	12.754	3.586	31.471
60	0.732	0.674	0.831	25.335	6.952	104.960
70	1.158	1.043	1.361	49.229	15.321	193.686
80	1.722	1.504	2.035	96.111	23.212	255.337
90	2.806	2.258	3.859	158.820	39.963	564.425

Table 5: Execution times (in secs) of compact LP vs row generation for balanced binary trees, over 50 random instances, with comonotone bivariates. $k = int(0.4n)$

	Compact LP			Row Generation		
n	Mean	Min	Max	Mean	Min	Max
10	0.019	0.018	0.022	0.105	0.047	0.181
15	0.047	0.042	0.085	0.455	0.115	0.915
20	0.086	0.071	0.133	1.497	0.291	3.901
25	0.143	0.117	0.189	6.246	0.332	24.139
30	0.225	0.178	0.294	24.819	1.343	158.307
35	0.325	0.276	0.393	89.551	0.800	724.588

Table 6: Execution times (in secs) of compact LP vs row generation for star graph, over 50 random instances, with comonotone bivariates. $k = int(0.4n)$

5.2.2. Effect of structure of the tree on execution times: We now analyse the execution times of our compact linear program and row generation for various tree structures. For $n = 30$

random variables, we generate 25 instances for each possible value of height of the tree (1 to $n-1$). The star tree has the lowest possible height (1) and the series graph has the maximum achievable height ($n-1$). All the univariate probabilities are set to p and for each instance, p is generated randomly. Figures 8 to 10 provide the execution times under comonotone bivariate probabilities and small, medium and large values of k ($k = 10, 15, 20$ respectively). The table of execution times is provided alongside.

	Compact LP			Row Generation		
Height	Mean	Min	Max	Mean	Min	Max
1	0.205	0.174	0.247	33.818	33.334	35.403
5	0.207	0.160	0.263	10.561	6.750	18.708
10	0.214	0.179	0.266	8.031	4.457	12.320
15	0.209	0.164	0.283	4.086	1.999	7.470
20	0.206	0.155	0.281	2.641	1.234	4.515
25	0.214	0.185	0.269	1.695	1.299	2.645
29	0.204	0.161	0.239	0.978	0.947	1.076

Figure 8 **Execution times (in secs) of compact LP and row generation for various tree heights, n=30, k =10**

	Compact LP			Row Generation		
Height	Mean	Min	Max	Mean	Min	Max
1	0.175	0.153	0.227	10.359	9.927	10.777
5	0.180	0.143	0.246	4.809	2.472	10.019
10	0.182	0.150	0.233	2.010	0.755	3.383
15	0.194	0.156	0.254	1.352	0.829	2.607
20	0.191	0.151	0.254	1.107	0.757	1.539
25	0.179	0.153	0.221	0.839	0.544	1.324
29	0.183	0.145	0.222	0.559	0.535	0.718

Figure 9 **Execution times (in secs) of compact LP and row generation for various tree heights, n=30, k =15**

	Compact LP			Row Generation		
Height	Mean	Min	Max	Mean	Min	Max
1	0.176	0.154	0.230	0.701	0.666	0.802
5	0.172	0.147	0.213	0.726	0.492	1.304
10	0.178	0.143	0.218	0.647	0.443	1.026
15	0.174	0.141	0.225	0.507	0.352	0.657
20	0.175	0.144	0.226	0.423	0.339	0.567
25	0.171	0.135	0.213	0.367	0.285	0.508
29	0.157	0.131	0.210	0.280	0.266	0.312

Figure 10 **Execution times (in secs) of compact LP and row generation for various tree heights, n=30, k =20**

We make the following observations:

1. Comparison between compact LP and row generation: The compact LP is found to be much faster than row generation always. Especially observe the table along side Figure 8, for a height of 1, where the mean time taken by row generation is 33.818 seconds while compact LP takes just 0.205 seconds on an average.

2. Performance as a function of tree height: For row generation,as the height of the tree grows, the execution time reduces (see Figures 8 to 10 and the tables provided alongside). For our compact linear program, there does not appear to be an obvious connection between the tree structure and the execution time. This could be a consequence of the overall number of constraints in the compact linear program remaining the same for a given k and n, irrespective of the tree structure.

3. Performance as a function of k: From the tables provided along with Figures 8 to 10, row generation scheme clearly takes longer execution times for lower values of k (mean times of $33.818, 10.359$ and 0.701 secs for $k = 10, 15, 20$ respectively). For compact LP, a strong trend is not observed.

5.3. Robustness of the Conditionally Independent Chow-Liu Tree

In this subsection we consider the probability distribution on $n = 4$ Bernoulli random variables provided in Chow and Liu [1968], Table 1. The univariate probabilities computed from the joint distribution in the original paper are $p_1 = p_2 = p_3 = 0.55, p_4 = 0.5$ while the bivariate probabilities are $p_{12} = 0.4, p_{13} = p_{14} = 0.3, p_{23} = 0.45, p_{24} = 0.25, p_{34} = 0.25$. Three trees that best approximate the provided joint distribution were also provided in the same paper. We present the trees themselves in Figure 12. The trees are equivalent in that the sum of mutual information encoded by the probability distributions on the edges of the trees is the same. We show how our bounds help in making a further selection among these three trees. Towards this, we report the following:

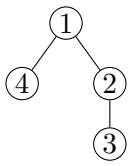
1. The probability $P_{ci} = \mathbb{P}(\sum_{i=1}^{n} \tilde{c}_i \geq k)$ assuming a conditional independent joint distribution on each of these trees (using Proposition 3). As the tree structures are different, the conditionally independent distributions themselves differ for these three trees.

2. Upper and lower bounds $U(k)$ and $L(k)$ on $\mathbb{P}(\sum_{i=1}^{n} \tilde{c}_i \geq k)$ assuming the bivariate distributions $\mathbb{P}(\tilde{c}_i = 1, \tilde{c}_j = 1)$ for all edges (i, j) in the given tree (using Theorem 2 and Equation (34) respectively). We also report the width $U(k) - L(k)$. This width represents the perturbation of probability bounds that can arise using a given tree.

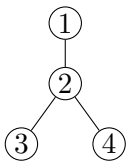
3. Upper and lower bounds $U_{uv}(k)$ and $L_{uv}(k)$ on $\mathbb{P}(\sum_{i=1}^{n} \tilde{c}_i \geq k)$ assuming the univariate probabilities $\mathbb{P}(\tilde{c}_i = 1)$ alone. $U_{uv}(k)$ was computed using Equation (4) while, by a similar argument in Equation (34), $L_{uv}(k) = U_{uv}(n-k+1)$ (where, in the computation of $U_{uv}(n-k+1)$ the probabilities $1 - p_i$ were sorted and used instead of p_i).

The univariate probabilities thus computed are provided in Table 7 (within Figure 12). These probabilities are the same for all the trees as they do not make use of any bivariate information.

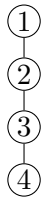
The range $[L_{uv}(k), U_{uv}(k)]$ is provided in the plots in Figure 12 using gray slanting lines. This band forms the widest band as the bounds assume only univariate information. Under assumptions of bivariate information on the trees, we see that the band $[L(k), U(k)]$ (plots depicted by the region shaded in red in Figure 12) is narrower as more information is assumed. These bands are clearly contained in the univariate band.



(a) Tree 1



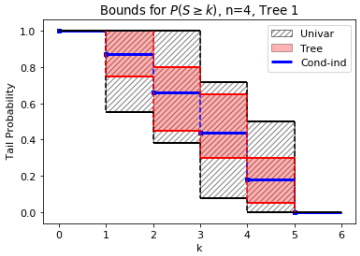
(b) Tree 2



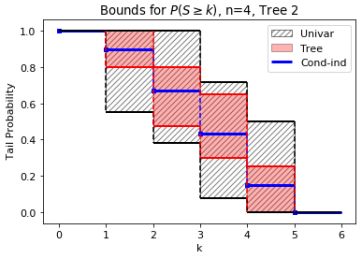
(c) Tree 3

k	$U_{uv}(k)$	$L_{uv}(k)$
1	1	0.55
2	1	0.3833
3	0.7167	0.30
4	0.5	0

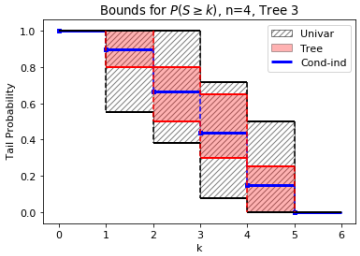
Table 7 Probability bounds assuming only univariate information



(a) Bounds for Tree 1



(b) Bounds for Tree 2



(c) Bounds for Tree 3

k	$U(k)$	$L(k)$	Width	P_{ci}
1	1	0.75	0.25	0.8704
2	0.8	0.45	0.35	0.6614
3	0.65	0.30	0.35	0.4397
4	0.3	0.05	0.25	0.1785

k	$U(k)$	$L(k)$	Width	P_{ci}
1	1	0.8	0.2	0.8963
2	0.8	0.475	0.325	0.6703
3	0.65	0.3	0.35	0.4346
4	0.25	0	0.25	0.1488

k	$U(k)$	$L(k)$	Width	P_{ci}
1	1	0.8	0.2	0.8963
2	0.8	0.5	0.3	0.6663
3	0.65	0.30	0.35	0.4386
4	0.25	0	0.25	0.1488

Figure 12 The trees in Chow and Liu [1968] and the bounds produced on $\mathbb{P}(\tilde{c}_i \geq k)$ under various assumptions of knowledge. All the three trees are characterised by equal mutual information from the true joint distribution.

Our key findings are the following:

1. Though the three trees are equivalent in terms of the mutual information, the range of probability bounds $[L(.), U(.)]$ differs for all of them. In every case, there is a significant benefit of using the bivariate information over univariate information. The univariate band is the same for all the trees.

2. In all cases, the bound from the conditionally independent distribution P_{ci} never matches with either $L(.)$ or $U(.)$.

3. For all values of k, tree 3 generates the lowest width $U(.) - L(.)$ while tree 1 generates the maximum width (see the tables (d), (e) and (f) in Figure 12). Hence, out of the three trees which are all optimal in terms of mutual information, Tree 3 gives the least perturbation in the bound.

6. Conclusions

We have proposed a polynomial time solvable linear program to compute upper and lower bounds for $\mathbb{P}(\sum_{i=1}^{n} c_i \geq k)$ for Bernoulli random variables, given bivariate information on a tree structure.

Our approach finds relevance especially in applications of vine copulas where bivariate information is only available on a tree structure. Our bounds are then useful in computing robust estimates. Our numerical results demonstrate the following aspects (1) Computational advantage of our compact linear program: Though the row generation scheme ultimately results in smaller sized formulations, the process of identifying the overall formulation is iterative and takes time. However the compact LP does not involve an iterative process and hence is faster. (2) Application of our compact LP in computing robustness estimates of Chow-Liu trees: Our compact LP distinguishes between the three trees optimal in terms of mutual information and can be used to choose the tree giving the least perturbation in bounds. (3) We derive insights on specific circumstances where including bivariate information gives more benefit over using only univariate information in terms of probability bounds.

Acknowledgments

This research was partly supported by the MOE Academic Research Fund Tier 2 grant T2MOE1706, "On the Interplay of Choice, Robustness and Optimization in Transportation". The authors would like to thank the editor Prof. Dmitris Bertsimas, the two anonymous reviewers and the associate editor for the detailed perusal of the paper and providing useful suggestions towards improving the paper.

References

Ahmed, S. and Papageorgiou, D. J. (2013). Probabilistic set covering with correlations. *Operations Research*, 61(2):438–452.

Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA.

Billionnet, A. (1992). Maximizing a quadratic pseudo-boolean function with a cardinality constraint. In *International Colloqium on Graphs and Optimization*.

Boland, P. J. and Proschan, F. (1983). The reliability of k out of n systems. *The Annals of Probability*, 11(3):760–764.

Bonami, P., Günlük, O., and Linderoth, J. (2018a). Globally solving nonconvex quadratic programming problems with box constraints via integer programming methods. *Mathematical Programming Computation*, 10(3):333–382.

Bonami, P., Lodi, A., and Zarpellon, G. (2018b). Learning a classification of mixed-integer quadratic programming problems. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 595–604.

Boros, E. and Prekopa, A. (1989). Closed form two-sided bounds for probabilities that at least r and exactly r out of n events occur. *Mathematics of Operations Research*, 14(2):317–342.

Boros, E., Scozzari, A., Tardella, F., and Veneziani, P. (2014). Polynomially computable bounds for the probability of the union of events. *Mathematics of Operations Research*, 39(4):1311–1329.

Brechmann, E. C., Czado, C., and Aas, K. (2012). Truncated regular vines in high dimensions with application to financial data. *Canadian Journal of Statistics*, 40(1):68–85.

Brechmann, E. C. and Joe, H. (2015). Truncation of vine copulas using fit indices. *Journal of Multivariate Analysis*, 138:19 – 33.

Bukszár, J. and Szántai, T. (2002). *Probability bounds given by hypercherry trees.* Optimization Methods and Software, 17(3):409–422.

Caprara, A., Pisinger, D., and Toth, P. (1999). Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–137.

Chang, B., Pan, S., and Joe, H. (2019). Vine copula structure learning via monte carlo tree search. In *Proceedings of Machine Learning Research*, volume 89, pages 353–361.

Chen, T. (1998). Optimal lower bounds for bivariate probabilities. *Advances in Applied Probability*, 30(2):476–492.

Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

Cooke, R. M., Joe, H., and Chang, B. (2019). Vine copula regression for observational studies. *AStA Advances in Statistical Analysis*, pages 1–27.

Dawson, D. and Sankoff, D. (1967). *An inequality for probabilities.* Proceedings of the American Mathematical Society, 18(3):504–507.

Dhara, A., Das, B., and Natarajan, K. (2019). Worst-case expected shortfall with univariate and bivariate marginals. *INFORMS Journal on Computing (To appear).*

Dissmann, J., Brechmann, E., Czado, C., and Kurowicka, D. (2013). Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, 59(C):52–69.

Embrechts, P. and Puccetti, G. (2010). Bounds for the sum of dependent risks having overlapping marginals. *Journal of Multivariate Analysis*, 101(1):177 – 190.

Fomeni, F. D. and Letchford, A. N. (2014). A dynamic programming heuristic for the quadratic knapsack problem. *INFORMS Journal on Computing*, 26(1):173–182.

Grötschel, M., Lovász, L., and Schrijver, A. (1988). Geometric algorithms and combinatorial optimization. *Algorithms and Combinatorics*, 2:65–84.

Hailperin, T. (1965). *Best possible inequalities for the probability of a logical function of events.* The American Mathematical Monthly, 72(4):343–359.

Hunter, D. (1976). *An upper bound for the probability of a union.* Journal of Applied Probability, 13(3):597–603.

Jr., D. J. R. and Woeginger, G. J. (2002). The quadratic 01 knapsack problem with seriesparallel support. *Operations Research Letters*, 30(3):159 – 166.

Kounias, E. G. (1968). *Bounds for the probability of a union, with applications.* The Annals of Mathematical Statistics, 39(6):2154–2158.

Kounias, S. and Marin, J. (1976). *Best linear Bonferroni bounds.* SIAM Journal on Applied Mathematics, 30(2):307–323.

Kurowicka, D. (2010). *Optimal Truncation of Vines*, pages 233–247.

Kwerel, S. M. (1975). *Most stringent bounds on aggregated probabilities of partially specified dependent probability systems*. Journal of the American Statistical Association, 70(350):472–479.

Lauritzen, S. (1996). *Graphical Models*. Oxford Statistical Science Series. Clarendon Press.

Morgenstern, D. (1980). Berechnung des maximalen signifikanzniveaus des testes lehneh0 ab, wennk untern gegebenen tests zur ablehnung fuhren. *Metrika*, 27(1):285–286.

Nelsen, R. B. (2006). *An Introduction to Copulas (Springer Series in Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Nemhauser, G. and Wolsey, L. (2014). *General Algorithms*, chapter II.4, pages 349–382. John Wiley and Sons, Ltd.

Padberg, M. (1989). The boolean quadric polytope: Some characteristics, facets and relatives. *Math. Program.*, 45(1):139–172.

Pitowsky, I. (1991). Correlation polytopes: Their geometry and complexity. *Mathematical Programming*, 50(1):395–414.

Prékopa, A. (1988). Boole-bonferroni inequalities and linear programming. *Operations Research*, 36(1):145–162.

Prékopa, A. (1990a). The discrete moment problem and linear programming. *Discrete Applied Mathematics*, 27(3):235 – 254.

Prékopa, A. (1990b). Sharp bounds on probabilities using linear programming. *Operations Research*, 38(2):227–239.

Prékopa, A. (2003). Probabilistic programming. *Handbooks in operations research and management science*, 10:267–351.

Prékopa, A. and Gao, L. (2005). Bounding the probability of the union of events by aggregation and disaggregation in linear programs. *Discrete applied mathematics*, 145(3):444–454.

Prékopa, A., Vizvári, B., and Regös, G. (1997). *A method of disaggregation for bounding probabilities of Boolean functions of events*. Rutgers University. Rutgers Center for Operations Research [RUTCOR].

Puccetti, G. and Wang, R. (2015). Extremal dependence concepts. *Statist. Sci.*, 30(4):485–517.

Qiu, F., Ahmed, S., and Dey, S. S. (2016). Strengthened bounds for the probability of k-out-of-n events. *Discrete Applied Mathematics*, 198:232–240.

Ruger, B. (1978). Das maximale signifikanzniveau des tests lehne ho ab, wenn k unter n gegebenen tests zur ablehnungfuhren. Metrika, 25:171–178.

Rüschendorf, L. (1991). Bounds for distributions with multivariate marginals. *Lecture Notes-Monograph Series*, pages 285–310.

Saxena, A., Goyal, V., and Lejeune, M. A. (2009). Mip reformulations of the probabilistic set covering problem. *Math. Program.*, 121(1):131.

Tomescu, I. (1986). Hypertrees and bonferroni inequalities. *Journal of Combinatorial Theory, Series B*, 41(2):209–217.

Vorobev, N. (1962). Consistent families of measures and their extensions. *Theory of Probability & Its Applications*, 7(2):147–163.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(12):1–305.

Wang, S. (1998). An actuarial index of the right-tail risk. *North American Actuarial Journal*, 2(2):88–101.

Worsley, K. (1982). An improved bonferroni inequality and applications. *Biometrika*, 69(2):297–302.

Yang, J., Alajaji, F., and Takahara, G. (2016). Lower bounds on the probability of a finite union of events. *SIAM Journal on Discrete Mathematics*, 30(3):1437–1452.