A shortest-path-based approach for the stochastic knapsack problem with non-decreasing expected overfilling costs

by

Troels Martin Range, Dawid Kozlowski and Niels Chr. Petersen

A shortest-path-based approach for the stochastic knapsack problem with non-decreasing expected overfilling costs

Troels Martin Range[*1,2], Dawid Kozlowski[3], and Niels Chr. Petersen[3]

[1]Hospital of South West Jutland and Institute of Regional Health Research: Centre of South West Jutland, University of Southern Denmark, Finsensgade 35, DK-6700 Esbjerg, Denmark
[2]Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Alfred Getz veg 3, NO-7491 Trondheim, Norway
[3]Department of Business and Economics, and COHERE, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark

July 4, 2017

Abstract

The knapsack problem (KP) is concerned with the selection of a subset of multiple items with known positive values and weights such that the total value of selected items is maximized and their total weight does not exceed capacity. Item values, item weights, and capacity are known in the deterministic case. We consider the stochastic KP (SKP) with stochastic item weights. We combine the formulation of the SKP with a probabilistic capacity constraint (CKP) and the SKP with simple recourse (SRKP). Capacity is made an integral part of the constraint set in terms of a chance constraint as in the CKP. The chance constraint allows for a violation of capacity, but the probability of a violation beyond an imposed limit is constrained. The capacity constraint is also included in the objective function in terms of a penalty function as in the SRKP. Penalty is an increasing function of the expected number of units of violation with proportionality as a special case. We formulate the SKP as a network problem and demonstrate that it can be solved by a label-setting dynamic programming approach for the Shortest Path Problem with Resource Constraint (SPPRC). We develop a dominance criterion for an elimination of states in the dynamic programming approach using only the deterministic value of items along with mean and variance of the stochastic weight of items corresponding to the associated paths in the underlying network. It is shown that a lower bound for the impact of potential extensions of paths is available as an additional means to limit the number of states provided the penalty cost of expected overtime is convex. Our findings are documented in terms of a computational study.

Keywords: Stochastic Knapsack Problem, Dynamic Programming, Shortest Path Problem with Resource Constraints

JEL code: C61, *MSC codes:* 90C15, 90C27, 90C35

---

[*]Corresponding author, e-mail: Troels.Martin.Range@rsyd.dk

1   Introduction

The starting point for the knapsack problem (KP) is a set of multiple items and a knapsack with a given capacity. Each item is associated with a positive value and a weight. The problem concerns the selection of which items to put in the knapsack such that the total value of the selected items is maximum, and their total weight does not exceed capacity. KP is the simplest non-trivial combinatorial optimization problem since the set of constraints is limited to a single capacity constraint and since all criterion coefficients are positive. The simplicity of the problem is one of the reasons why it has been given considerable attention in academia, as it may be possible to generalize the solution techniques for a simple model to a more complex model. In addition, an iterative solution of the KP or variants of it is in some instances an integral part of the solution method for a more complex problem. We refer to Kellerer et al. (2004) for a comprehensive treatment of the knapsack problem along with a large number of its variants.

The formulation and solution of the surgery allocation problem for a hospital's operating theater suggested by Range et al. (2016) is an example of a complex model with a variant of the KP employed as an integral part of the problem solution. Surgery allocation is concerned with the dynamic allocation of already arrived patients and sufficient planned slack in terms of tentative slots to accommodate the allocation of future patient arrivals to an available surgeon and day for surgery before a due date. The objective is to maximize capacity utilization and minimize the cost of overfilling operating rooms in terms of physicians' overtime and the cost of referral of arriving patients to other hospitals, because the cost of offering a patient surgery before the due date outweighs the benefits. This problem can be decomposed into a generalized assignment problem (GAP) and a stochastic knapsack problem (SKP) with an increasing convex cost function on the expected overfill cost.

The surgery allocation problem was the starting point for the current study. Focus is on a static SKP with the set of items given a priori and with stochastic item weights. The combination of a given surgeon in an operating room on a given day defines a knapsack with a capacity determined by the surgeon's working hours along with the room's opening hours. The problem is to allocate patients – already arrived or to arrive in the future – to feasible combinations of surgeons and days such that the corresponding operating room is utilized in the best possible way. Accordingly, on the one hand we do not want the surgeon to be idle, while on the other hand we do not want to overfill the room. Patients define the set of items from which a subset is to be selected. The deterministic value of an item is measured by the revenue obtained by the allocation of a patient to surgery. The weight of an item is the duration of surgery, which is stochastic. The mean and the variance for the size of each item is given, but not necessarily the exact distribution. However, the distribution of the duration for completing all surgeries allocated to a given surgeon on a given day is approximately normal by the central limit theorem, and the mean and the standard deviation for this distribution can be computed. An upper limit on the total weight of selected items is an integral part of the constraint set, and a recourse imposing a penalty for violating this limit is embedded in the criterion function in terms of a function that is a non-decreasing function of the expected violation.

The formulation of the SKP in this study allows for an incorporation of additional constraints. Additional constraints may, for example, impose the requirement that the accumulated use of other resources than knapsack capacity must not exceed their availability, that the set of items is partitioned into subsets and only one item in each subset is allowed to be used, or that the total number of selected items must not exceed an imposed upper bound. Constraints of this type are common in practice but may render

the solution procedures for the standard knapsack void. It is shown that the variant of the SKP to be developed in this paper can be formulated as a network problem and solved by a label-setting dynamic programming algorithm for the shortest path problem with resource constraints (SPPRC), which allows for an incorporation of additional constraints such as the ones mentioned above.

The remainder of the paper unfolds as follows: In Section 2 we briefly describe related literature. In Section 3 we give the details on the model we are solving, and in Section 4 we describe the network-based solution approach. This is followed by a computational study in Section 5. Finally, concluding remarks and potential future developments are discussed in Section 6. Proofs of the propositions are given in the appendix.

2 Review

Item weights and revenues are known parameters in a deterministic knapsack problem (KP) as is the capacity of the knapsack. The problem is turned into a stochastic optimization problem when these parameters are stochastic and modeled as random variables with certain distributional characteristics. The stochastic knapsack problem (SKP) is $\mathcal{NP}$-hard as is the underlying deterministic version of the SKP.

The class of stochastic knapsack problems can be classified as dynamic or static.[1] Items are put in the knapsack one by one in the dynamic version of the problem, and the stochastic weight and/or revenue becomes known when an item has been chosen to be included in the knapsack. Accordingly, the dynamic version can take advantage of feedback from the system, since the current load and/or revenue is known with certainty as is the incremental impact of the next item. By contrast, the decision of the complete set of items to be put in the knapsack is made before the stochastic parameters become known in a static SKP. There is no feedback from the system, and a simultaneous disclosure of the realization of all stochastic parameters determines the outcome of an equally simultaneous placing of all items in the knapsack.

İlhan et al. (2011) consider the dynamic SKP in the scenario with stochastic revenues and deterministic weights along with a maximization of the probability for a realization of a predetermined revenue target. The distinquishing feature of this scenario is that capacity is not violated, since weights are known with certainty. Derman et al. (1978) and Dean et al. (2008) address the scenario with random weights and deterministic revenues. The capacity of the knapsack is in this case violated whenever the realization of the random weight for the next item to be put into knapsack implies a total weight exceeding capacity. The objective is a maximization of total revenue without violating capacity or breaking the knapsack.

This paper is concerned with the static SKP. The static SKP can be classified into three categories. SKPs in category 1 consider the scenario with stochastic revenues and deterministic weights. Focus is on a maximization of the probability for obtaining a target for revenue while satisfying capacity. Hence, this type of SKPs belong to the class of so-called stochastic target achievement problems since the objective is a maximization of the probability for a realization of an outcome of a stochastic process. Henig (1990), Morton and Wood (1998), Steinberg and Parks (1979), and Sniedovich (1980) are examples. Furthermore, Merzifonluoğlu et al. (2012) discusses the static SKP with normally distributed weights with a linear penalty on violating the capacity constraint.

---

[1]Chen and Ross (2015) call the classification adaptive vs. static.

SKPs in categories 2 & 3 consider the scenario with random weights. Specific assumptions are usually not imposed since focus is on a maximization of expected revenue and since mean revenue values are sufficient for that purpose.

Category 2 SKPs maximize expected revenue given that the probability for violating capacity is not allowed to exceed a certain level to be specified a priori. The transformation of a deterministic capacity constraint into the requirement that the probability for a violation must not exceed a certain level classifies the problem as a so-called (chance) constrained knapsack problem (CKP). Kosuch and Lisser (2009) consider the CKP.

Category 3 SKPs also maximize expected utility. The distinguishing feature is that the fulfillment of the capacity constraint is addressed in the criterion. To be more specific, the criterion includes a penalty for a violation of capacity, which in turn gives rise to the SKP with recourse (SRKP). The penalty cost is usually assumed to be proportional to the level of violation of capacity; see, e.g., Kleywegt and Papastavrou (2001). The SRKP includes the broken knapsack problem (BKP) as a special case, where revenue drops to zero if capacity becomes violated. Hence, expected revenue to be maximized equals revenue generated by the value of the items in the knapsack multiplied by the probability that capacity is not violated. We refer to Kosuch and Lisser (2009) for a brief review of the CKP and the SRKP.

The solution of the CKP is based on chance constrained programming. The underlying idea is that the probabilistic capacity constraint under certain distributional assumptions for the stochastic weights can be replaced by a deterministic equivalent. The resulting optimization problem is in general non-linear and can be solved by a convex optimization method. The stochastic gradient method provides a possible solution approach for the SRKP. An optimal solution for the SKP can also be obtained by a Branch & Bound algorithm, since the problem is concerned with combinatorial optimization; see, e.g., Carraway et al. (1993). Kleywegt et al. (2002) study the sample average approximation method for stochastic discrete optimization and apply their findings to the static SKP. The sample average method is appropriate in scenarios where a stochastic criterion cannot be written in closed form or cannot be easily calculated. The SRKP is one example of this.

Tönissen et al. (2017) addresses a slightly different version of the SKP called the two-stage stochastic multiple knapsack problem. The problem is static in the sense that multiple knapsacks each one with a given capacity must be filled with items each of which is associated with a positive value and a deterministic weight in stage one. The problem is dynamic in the sense that there is a (small) probability that the a priori known capacity of any given knapsack may be reduced in stage two due to a stochastic requirement that additional items must be included in the knapsack. The stochastic inclusion of additional items in stage two necessitates the removal of items placed in the knapsack in stage one, and the problem is concerned with the maximization of the total expected value of items included in the knapsacks at the end of stage two. The problem is easily seen to relate to the surgery allocation problem in an operating theater, where knapsacks correspond to operating rooms and the stochastic inclusion of new items in stage two to the arrival of emergency patients. The suggested solution algorithms involve column generation and decomposition procedures.

The version of the static SKP to be addressed in this paper belongs to a combination of the CKP and the SRKP. We maximize expected profits in terms of revenue minus expected cost of a capacity violation. This is similar to the SRKP. The maximization is subject to a constraint stating that the expected value of the random variable measuring total weight plus a predefined number of standard deviations

for this variable is not allowed to exceed an imposed upper bound. This is similar to the CKP, since the requirement basically implies that the probability that the random variable total weight exceeds the upper bound is not allowed to exceed a certain level.

The model is a non-linear combinatorial optimization problem. It is shown that the problem can be solved as a shortest path problem with resource constraints (SPPRC) on a directed graph. The use of resource extension functions allows for the development of a dynamic programming algorithm. We add to the literature on the static SKP by relaxing the standard assumption of proportionality between penalty cost and capacity violation. To be more specific, we model penalty as an increasing function of expected overfill. The solution of the dynamic programming problem is based on a labeling approach reflecting states of the underlying system. The efficiency of the labeling procedure increases if the number of potential states can be reduced based on dominance criteria. We also add to the literature on the static SKP by the development of bounds for the criterion values corresponding to potential states provided penalty is an increasing and convex function in expected overfill. The advantages of the bounding procedure are documented in terms of a computational study.

3   The extended stochastic knapsack model

Suppose that we are given the set of *independent* items, $\mathcal{J} = \{1, \ldots, J\}$, and with each item a revenue, r_j, as well as an expected weight, $\mu_j = \mathbb{E}[X_j] > 0$, and variance, $\sigma_j^2 = \mathbb{V}[X_j] \geq 0$, for the stochastic variable, X_j. Certainty regarding item weight prevails if the variance is zero, and the problem becomes deterministic if the variance is zero for all items. An upper limit, T, on the total weight of the selected items is given. The expected amount with which the total weight is violated is penalized by a non-decreasing function, $f : \mathbb{R}_+ \to \mathbb{R}$, having $f(0) = 0$. Furthermore, a maximum amount, S, of violation is allowed such that the probability that the weight $T + S$ is surpassed does not exceed a predefined level.

For each element $j \in \mathcal{J}$ we let $x_j \in \{0, 1\}$ be a binary variable indicating whether or not the element is included in the solution. We will denote a solution $\overline{\mathbf{x}} = (\overline{x}_1, \ldots, \overline{x}_J)$. It is assumed that the stochastic variables, X_j, are independent, and as a consequence, we can rely on the central limit theorem to obtain an approximation of the distribution. The amount in the knapsack corresponds to the realization of the stochastic variables of the elements included (i.e., $\overline{X} = \overline{x}_1 X_1 + \ldots, \overline{x}_J X_J$), and we can measure the mean fill of the knapsack for a solution, $\overline{\mathbf{x}}$, as

$$\overline{\mu} = \sum_{j \in \mathcal{J}} \mu_j \overline{x}_j$$

and the standard deviation as

$$\overline{\sigma} = \sqrt{\sum_{j \in \mathcal{J}} \sigma_j^2 \overline{x}_j}$$

We have by the central limit theorem that $\overline{X} = \sum_{j \in \mathcal{J}} \overline{x}_j X_j$ is approximately normally distributed with mean $\overline{\mu}$ and standard deviation $\overline{\sigma}$.

3.1   Approximation of the expected overfill

We are interested in identifying the expected overfilling of the knapsack as the cost, f, depends on the expected overfilling. We denote $O = \max\{0; X - T\} = (X - T)^+$ as the stochastic variable measuring

5

the amount of overfill. The following proposition states an approximation of the expected overfill $\mathbb{E}[O]$:

Proposition 1. *Let $X_1, \ldots, X_n$ be a set of independent stochastic variables with means μ_i and variances σ_i^2 for $i = 1, \ldots, n$. Let $X = X_1 + \ldots + X_n$ and $O = (X - T)^+$ for a constant $T \geq 0$. Denote $\mu_X = \mu_1 + \ldots + \mu_n$ and $\sigma_X^2 = \sigma_1^2 + \ldots + \sigma_n^2$. Then*

$$\mathbb{E}[O] \approx \sigma_X \left(\phi(k) - k(1 - \Phi(k)) \right)$$

where $\phi(\cdot)$ is the probability density function and $\Phi(\cdot)$ is the cumulative distribution function for the standard normal distribution and $k = (T - \mu_X)/\sigma_X$.

Kleywegt et al. (2002) observe without proof a similar result in the case where the X variables are normally distributed. We will denote the approximation of the expected overfill based on μ and σ when the knapsack has capacity T as

$$h_T(\mu, \sigma) = \sigma \left[\phi \left(\frac{T - \mu}{\sigma} \right) - \frac{T - \mu}{\sigma} \left(1 - \Phi \left(\frac{T - \mu}{\sigma} \right) \right) \right] \tag{1}$$

which corresponds to the approximation given i Proposition 1. An illustration of this approximation along with $\max\{0, \mu - T\}$ is given in Figure 1. In the figure we have fixed σ at either of the four levels 10, 20, 50, and 100. Below we will formally argue that $h_T(\mu, \sigma)$ is increasing in both μ and σ and $\max\{0; \mu - T\}$ serves as a lower bound for $h_T(\mu, \sigma)$ when we decrease σ. The first order derivatives of $h_T(\mu, \sigma)$ are
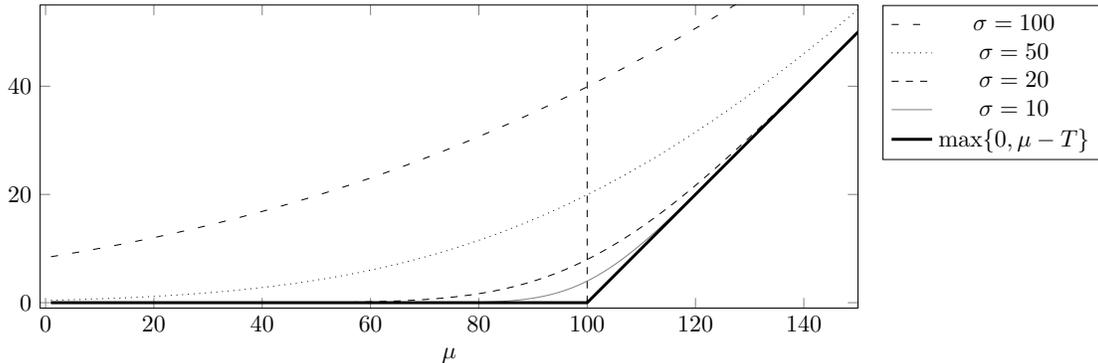

Figure 1: Illustration of $h_T(\mu, \sigma)$ with $\sigma = 10, 20, 50, 100$ and $\max\{0, \mu - T\}$ for $T = 100$.

stated in Proposition 2.

Proposition 2. *Let $\mu \geq 0$ and $T \geq 0$, and let $\phi(\cdot)$ be the probability density function and $\Phi(\cdot)$ be the cumulative distribution function for the standard normal distribution, and let $h_T(\mu, \sigma)$ be defined as in (1). Then*

$$\frac{\partial h_T(\mu, \sigma)}{\partial \mu} = 1 - \Phi \left(\frac{T - \mu}{\sigma} \right)$$

and

$$\frac{\partial h_T(\mu, \sigma)}{\partial \sigma} = \phi \left(\frac{T - \mu}{\sigma} \right)$$

An important consequence of Proposition 2 is that $h_T(\mu, \sigma)$ is strictly increasing, as $\phi(k) > 0$ and $\Phi(k) < 1$ for all k. Intuitively, this is not surprising, as we would expect the overtime to increase when we fill up the

knapsack more as well as when the uncertainty increases. We will exploit this when setting up the solution approach for the problem. We can also obtain a deterministic lower bound for the function $h_T(\mu, \sigma)$ (i.e., where the standard deviation approaches zero and therefore the mean becomes the deterministic value of the weight). This bound is given in Proposition 3.

Proposition 3. *Let $h_T(\mu, \sigma)$ be defined as in (1) and $\mu \geq 0$ be an arbitrary mean value. Then*

$$h_T(\mu, \sigma) \quad \geq \quad \max\{0; \mu - T\}$$

and $h_T(\mu, \sigma) \searrow \max\{0; \mu - T\}$ monotonically for $\sigma \searrow 0$.

Proposition 3 fits the intuition that if we have no uncertainty about the weight of the items, then the expected overfill will be zero if the fill, μ, is below T, while it will be $\mu - T$ if μ is above T. We will use the lower bound given in Proposition 3 to accelerate the solution process.

3.2 The maximum overfill

From a managerial perspective it is of interest not to violate the limit $T + S$. However, as the item weights are stochastic we can only guarantee this with a certain probability. Observe the following constraint:

$$\sum_{j \in \mathcal{J}} \mu_j x_j + \beta \sqrt{\sum_{j \in \mathcal{J}} \sigma_j^2 x_j} \leq T + S \tag{2}$$

where $\beta \geq 0$ corresponds to the number of standard deviations we are interested in having as slack from the mean and up to $T + S$. As the distribution is approximately normally distributed, this translates to the probability of having a value no larger than $T + S$. The assumption that $\beta \geq 0$ reflects the fact that we would like the fill of the knapsack to be less than or equal to $T + S$ with a probability of at least 50% (corresponding to the case where $\beta = 0$). When β is increased, then so is the corresponding probability of the fill being no larger than $T + S$. Indeed, this corresponds to a chance constraint stating that the probability of surpassing $T + S$ for the sum of the weights of the individual items should not be larger than a predefined value.

3.3 Disjoint elements

In some applications the items of $\mathcal{J}$ are partitioned into K groups of items in which maximally one item can be chosen. Let $\mathcal{K} = \{1, \ldots, K\}$ be the index set of the groups and let $\mathcal{J}_k \subseteq \mathcal{J}$ with $\mathcal{J} = \cup_{k \in \mathcal{K}} \mathcal{J}_k$ and $\mathcal{J}_k \cap \mathcal{J}_h = \emptyset$ for $h, k \in \mathcal{K}$ and $h \neq k$. Then maximally one element from each set, $\mathcal{J}_k$, can be chosen which corresponds to

$$\sum_{j \in \mathcal{J}_k} x_j \leq 1, \quad k \in \mathcal{K} \tag{3}$$

If $|\mathcal{J}_k| = 1$ for all $k \in \mathcal{K}$, then the problem is a binary-type KP. An important special case of this is where the number of items we can select of the same type is not restricted to being binary, but rather integer with some upper bound (i.e., $0 \leq x_j \leq u_j$ and $x_j \in \mathbb{Z}$). In this case we construct u_j new items, x_{aj}, corresponding to having $a = 1, \ldots, u_j$ items, with $r_{aj} = ar_j$, $\mu_{aj} = a\mu_j$, and $\sigma_{aj}^2 = a\sigma_j^2$. These new items are then collected into one group, k, such that maximally one of them can be used. Consequently, we can have an integer-type SKP.

3.4 The model

Using Proposition 1 we can approximate the expected overfilling $\mathbb{E}[O] \approx h_T(\mu_X, \sigma_X)$ and thereby set up the objective

$$\sum_{j \in \mathcal{J}} r_j x_j - f(h_T(\mu_X, \sigma_X)) \tag{4}$$

which is to be maximized. We can now state the SKP as

$$\max \quad \sum_{j \in \mathcal{J}} r_j x_j - f\left(h_T\left(\sum_{j \in \mathcal{J}} \mu_j x_j, \sqrt{\sum_{j \in \mathcal{J}} \sigma_j^2 x_j} \right) \right) \tag{5}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{J}_k} x_j \quad\quad\quad\quad\quad \leq \quad 1, \quad\quad k \in \mathcal{K} \tag{6}$$

$$\sum_{j \in \mathcal{J}} \mu_j x_j + \beta \sqrt{\sum_{j \in \mathcal{J}} \sigma_j^2 x_j} \quad\quad \leq \quad T + S \tag{7}$$

$$x_j \in \{0, 1\}, \quad\quad\quad\quad\quad\quad\quad\quad j \in \mathcal{J} \tag{8}$$

The objective (5) maximizes the profit by the difference in revenue and expected cost. The first set of constraints, (6), ensures that maximally one element from each partition, $\mathcal{J}_k$ with $k \in \mathcal{K}$, is chosen, and that it corresponds to constraint (3). Given that we can calculate μ_X and σ_X, then (7) ensures that the mean fill will be at least β standard deviations below the strict upper bound of $T + S$, which is indeed equivalent to (2). Finally, the domain of the x_j variable is given in (8). The model (5)-(8) is inherently non-linear. Fortunately, the structure of the model lends itself to a network-based approach, which we will discuss in Section 4.

4 A shortest path based approach

The problem given in Section 3 can be formulated as an SPPRC on a special directed graph. In this section we will first set up the graph and then we will describe how to formulate the elements of the SKP by applying resource extension functions. This leads to a dynamic programming algorithm on the graph, where dominance is necessary only on the cost function, the mean, the variance, and the potential side constraints, but where it is not necessary to apply dominance on the expected overfill. We close this section by discussing a bound that can be used to eliminate potential states from the dynamic programming.

The shortest path problem with time windows was described by Desrochers and Soumis (1988), who gave a permanent label setting (dynamic programming) algorithm based on the observation that time is strictly increasing and can be used to make extensions where the extended state would never be eliminated. In effect, they divide time into buckets no larger than the minimum amount of time required for an extension and then conduct the extension in increasing order of the bucket indices. While time is an important part of many shortest path problems, it can be generalized into so-called resource constraints. Desaulniers et al. (1998) introduced the notion of resource extension functions (REFs) which are functions taking a resource state and computing the consequence on the resource state when it is extended along an arc. The authors show that if the REFs are non-decreasing and separable then a standard dominance

criterion can be applied for elimination of states. Irnich (2008) further elaborates on the characteristics and properties of REFs. A general description of SPPRCs is given by Irnich and Desaulniers (2005).

Much of the development of SPPRC algorithms has been part of other related problems. Desrochers et al. (1992) embedded SPPRC in a column generation approach to solve the Vehicle Routing Problem with Time Windows. This led to the so-called Elementary SPPRC where paths are prohibited from having cycles even though the underlying network contains cycles and dynamic programming algorithms have been developed, see for instance Feillet et al. (2004), Chabrier (2006), and Righini and Salani (2006) and Lozano and Medaglia (2013). Gauvin et al. (2014) solves a vehicle routing problem with stochastic demands by column generation. Indeed they solve a related shortest path problem related to the one we apply, where the cumulative expected demand and cumulative variance are discretized to create a state-space graph. We do not apply a discretization in our solution approach. Shortest path problems are also used in the context of staff scheduling or rostering where the underlying graph is acyclic as described by for example Eveborn and Rönnqvist (2004), Lusby et al. (2012), and Smet et al. (2016). Below we will focus on the dynamic programming approach which can be derived from a shortest path formulation of the stochastic knapsack problem.

As we are going to use a shortest path algorithm we will minimize the negative profit function instead of maximizing the profit function directly. Thus, the objective we minimize is

$$f(h_T(\mu_X, \sigma_X)) - \sum_{j \in \mathcal{J}} r_j x_j \tag{9}$$

which corresponds to (4) multiplied by -1, and the resulting objective value of this minimization should also be multiplied by -1 to obtain the profit.

The graph can be constructed as follows. First denote $\mathcal{V} = \{o, d\} \cup \mathcal{J}$ as the set of nodes or vertices of the graph, where $o = 0$ corresponds to the origin and $d = J + 1$ to the destination, and the remaining nodes correspond to each of the elements of $\mathcal{J}$. The origin, o, and destination, d, have no associated revenue, mean, or variance and we put $r_o = r_d = 0$, $\mu_o = \mu_d = 0$, and $\sigma_o^2 = \sigma_d^2 = 0$.

The set of arcs is $\mathcal{A} \subseteq \{(i, j) \in \mathcal{V} \times \mathcal{V} | i < j\}$. The resulting graph will be acyclic, as all arcs have a tail-node index that is lower than the head node index. With each arc $(i, j) \in \mathcal{A}$ we associate the simple accumulation of revenue $r_{ij} = r_j$, mean $\mu_{ij} = \mu_j$, and variance $\sigma_{ij}^2 = \sigma_j^2$, corresponding to the values in the head node. All arcs (i, j) with $i, j \in \mathcal{J}_k$ with $k \in \mathcal{K}$ are excluded from the set of arcs, $\mathcal{A}$, which makes it impossible to use two items from the same subset, $\mathcal{J}_k$. Furthermore, we let $x_{ij} \in \{0, 1\}$ indicate whether or not arc (i, j) is used in a solution.

An example of a $\mathcal{J} = \{1, \dots, 5\}$ item graph is illustrated into Figure 2. $\mathcal{J}$ is partitioned in the two subsets $\mathcal{J}_1 = \{1, 2\}$ and $\mathcal{J}_2 = \{3, 4, 5\}$, where $\mathcal{K} = \{1, 2\}$. These sets are illustrated by the dashed boxes. Arcs from o and to d are faded in order to ease the exposition.

A path $\mathcal{P} = (o, v_1, \dots, v_p, d)$ from the origin, o, to the destination, d, in the graph is a connected sequence of arcs $(v_q, v_{q+1}) \in \mathcal{A}$ for $q = 0, \dots, p$, where $v_0 = o$ and $v_{p+1} = d$. We write that an arc $(i, j) \in \mathcal{P}$ if the path traverses the arc and, likewise, that $i \in \mathcal{P}$ if the path visits node i. A path, $\mathcal{P}$, translates into a solution for the KP by $x_{ij} = 1$ if and only if $(i, j) \in \mathcal{P}$ and

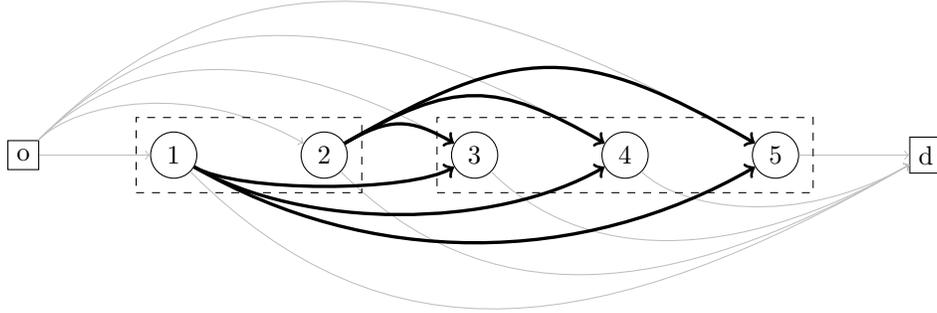$$x_j = \sum_{(i,j) \in \mathcal{A}} x_{ij} \tag{10}$$

Figure 2: Example of graph

Thus, if we can identify a path through this graph satisfying a possible set of side constraints, then we have a knapsack solution. Observe that due to the exclusion of arcs between nodes from the sets $\mathcal{J}_k$, a path will automatically satisfy the constraints (6).

We solve the KP by repeated extension of paths to successor nodes, where the successor nodes of $i \in \mathcal{V}$ are defined as $Succ(i) = \{j \in \mathcal{V} | (i,j) \in \mathcal{A}\}$. That is, we initialize the problem with a singleton path, $\mathcal{P} = (o)$, and extend it to all possible successor nodes to obtain the paths (o, v_1), where $v_1 \in \mathcal{J}$. Then each of these is extended to its successor nodes to obtain paths (o, v_1, v_2). This is repeated until all paths have been extended to all their possible successors. However, the number of potential paths increases exponentially with the size of K and can be bounded by $O(2^K)$, and as a consequence caution needs to be taken to avoid this. Indeed, not all paths are valid due to the maximum fill bound (2) and not all paths are good with respect to the objective.

With a partial path $\mathcal{P} = (o, v_1, \ldots, v_p)$ we will associate a state, $\mathcal{S}(\mathcal{P})$, of the path. The state corresponds to the accumulated revenue, mean, and variance, i.e. $\mathcal{S}(\mathcal{P}) = (r(\mathcal{P}), \mu(\mathcal{P}), \sigma^2(\mathcal{P}))$, where

$$r(\mathcal{P}) = \sum_{(i,j) \in \mathcal{P}} r_{ij}, \quad \mu(\mathcal{P}) = \sum_{(i,j) \in \mathcal{P}} \mu_{ij}, \quad \sigma^2(\mathcal{P}) = \sum_{(i,j) \in \mathcal{P}} \sigma^2_{ij}$$

and, based on this, the cost, which is slightly more complicated, is calculated as

$$C(\mathcal{P}) = f\left(h_T\left(\mu(\mathcal{P}), \sqrt{\sigma^2(\mathcal{P})}\right)\right) - r(\mathcal{P})$$

Note that the cost, $C(\mathcal{P})$, depends on the revenue, $r(\mathcal{P})$, the mean, $\mu(\mathcal{P})$, and the variance, $\sigma^2(\mathcal{P})$, of the path and it is therefore not separable from these. As it is not separable and it is the cost we are minimizing, the proof of the validity of the standard componentwise dominance criterion given by Desaulniers et al. (1998) does not apply directly to our situation. We will, however, see that the componentwise dominance criterion is valid in our case, too.

While we can calculate the mean, variance, and cost of a given path directly, it is practical to have simple ways of calculating these values when we extend a path to a new node. We assume that the path $\mathcal{P} = (o)$ has $C(\mathcal{P}) = \mu(\mathcal{P}) = \sigma^2(\mathcal{P}) = 0$, as it corresponds to an empty knapsack solution. If we extend

the path $\mathcal{P}$ with the node v_{p+1} to obtain the path $\mathcal{Q} = (o, v_1, \ldots, v_p, v_{p+1})$, then we have to identify the state $\mathcal{S}(\mathcal{Q})$. The extended value of the revenue, the mean, and the variance can be calculated by the following recursions:

$$r(\mathcal{Q}) = r(\mathcal{P}) + r_{v_p v_{p+1}} \tag{11}$$

and

$$\mu(\mathcal{Q}) = \mu(\mathcal{P}) + \mu_{v_p v_{p+1}} \tag{12}$$

and

$$\sigma^2(\mathcal{Q}) = \sigma^2(\mathcal{P}) + \sigma^2_{v_p v_{p+1}} \tag{13}$$

whereas the cost is calculated by the recursion

$$C(\mathcal{Q}) = C(\mathcal{P}) - r_{v_p v_{p+1}} + f\left(h_T\left(\mu(\mathcal{Q}), \sqrt{\sigma^2(\mathcal{Q})}\right)\right) - f\left(h_T\left(\mu(\mathcal{P}), \sqrt{\sigma^2(\mathcal{P})}\right)\right) \tag{14}$$

i.e., as the cost of path $\mathcal{P}$ from which the gained revenue has been subtracted and to which the increase in cost of expected overfill has been added.

An extension is feasible if it satisfies (6) and (7). Constraint (6) is trivially satisfied due to the construction of the graph. However, constraint (7) has to be checked. We observe that $\mu(\mathcal{Q}) > \mu(\mathcal{P})$ and $\sigma^2(\mathcal{Q}) \geq \sigma^2(\mathcal{P})$ due to $\mu_{v_p v_{p+1}} > 0$ and $\sigma^2_{v_p v_{p+1}} \geq 0$, respectively, and as a consequence

$$\mu(\mathcal{P}) + \beta\sqrt{\sigma^2(\mathcal{P})} < \mu(\mathcal{Q}) + \beta\sqrt{\sigma^2(\mathcal{Q})}$$

for any non-negative value β, and the extension of a path will result in increasing value on the left-hand side of (7). Hence, if $\mu(\mathcal{Q}) + \beta\sqrt{\sigma^2(\mathcal{Q})} > T + S$, then the left-hand side of (7) is violated for path $\mathcal{Q}$ and, more importantly, violated for any extension of $\mathcal{Q}$. Therefore, the path $\mathcal{Q}$ and all its extensions will be infeasible. We therefore say that path $\mathcal{P}$ is *not* extendable to the node v_{p+1}.

A reduction of the number of states is important to limit the combinatorial explosion of the algorithm. It is of particular interest to be able to compare paths in order to eliminate those paths for which better paths exist. A simple dominance relation is given in Proposition 4.

Proposition 4. *Let $\mathcal{P}_1, \mathcal{P}_2$ be two paths from o to node i. If*

1. *$r(\mathcal{P}_1) \geq r(\mathcal{P}_2)$,*

2. *$\mu(\mathcal{P}_1) \leq \mu(\mathcal{P}_2)$, and*

3. *$\sigma^2(\mathcal{P}_1) \leq \sigma^2(\mathcal{P}_2)$,*

then for any feasible extension of path $\mathcal{P}_2$ a corresponding feasible extension of path $\mathcal{P}_1$ exists yielding no larger variance, mean, or cost.

Proposition 4 states that path $\mathcal{P}_2$ can never be better than the path $\mathcal{P}_1$. This makes intuitive sense, as $\mathcal{P}_1$ gets a higher revenue with less, but more certain, filling of the knapsack than $\mathcal{P}_2$. We will say that the state $\mathcal{S}(\mathcal{P}_1)$ (weakly) dominates state $\mathcal{S}(\mathcal{P}_2)$ and we will write $\mathcal{S}(\mathcal{P}_1) \preceq \mathcal{S}(\mathcal{P}_2)$. Clearly, in the case that $\mathcal{S}(\mathcal{P}_1) \preceq \mathcal{S}(\mathcal{P}_2)$ we can always obtain a solution by extending $\mathcal{P}_1$, which is at least as good as extending $\mathcal{P}_2$, and we will therefore remove $\mathcal{P}_2$ from consideration.

It is interesting to note that the dominance criterion in Proposition 4 does not depend on the shape of the overfill cost function, f, as long as it is non-decreasing. As a consequence, we can embed the solution of this type of KP in an SPPRC framework.

The extension of paths and their states gives rise to an algorithm where we consider each node in the graph $(\mathcal{V}, \mathcal{A})$ once and extend all undominated paths from that node to all possible successor nodes. The pseudo code for the dynamic programming is given in Algorithm 1. The algorithm is generic in the sense that it applies in general to shortest path problems on directed acyclic graphs. However, it does not apply to the case where cycles are present.

Algorithm 1: Label setting dynamic programming algorithm

---

1 $\Lambda_i = \emptyset$;

2 $\Lambda_o = \{\mathcal{S}((o))\}$;

3 **foreach** $i \in \mathcal{V}$ *in topological order* **do**

4   **foreach** $\mathcal{S}(\mathcal{P}) \in \Lambda_i$ **do**

5   **foreach** $j \in Succ(i)$ **do**

6   **if** *extendable($\mathcal{S}(\mathcal{P})$, j)* **then**

7 $\mathcal{S}(\mathcal{Q}) = \text{extend}(\ \mathcal{S}(\mathcal{P}), j\ )$;

8   **if** *not dominated($\Lambda_j, \mathcal{S}(\mathcal{Q})$)* **then**

9   eliminate_ineffecient($\Lambda_j, \mathcal{S}(\mathcal{Q})$);

10   insert($\Lambda_j, \mathcal{S}(\mathcal{Q})$);

11   **end**

12   **end**

13   **end**

14   **end**

15 **end**

We denote by Λ_i all the undominated states present in node $i \in \mathcal{V}$. These sets are initialized to be empty, except for the origin in which the initial state of the path, $\mathcal{P} = (o)$, is inserted. Then each of the nodes in $\mathcal{V}$ is processed in topological order. Due to the topological order it will never be possible to extend states to a node from which we have already extended states. Consequently, when extending states from Λ_i, we know that the states included in Λ_i constitute the final pareto efficient set of states for the subset of nodes with indices less than or equal to node i. All the undominated states in node i are extended to successor nodes, j, if possible, and the dominance check is commenced to identify whether or not any state previously extended to node j dominates the newly generated state. The check of whether or not state $\mathcal{S}(\mathcal{P})$ is extendable to node j corresponds to checking constraint (7), while the extended state $\mathcal{S}(\mathcal{Q})$, when extending $\mathcal{S}(\mathcal{P})$ to node j, corresponds to calculating (11)-(13) as well as the cost (14). If the state is undominated, then it is used to eliminate other previously generated states after which it is inserted into the set, Λ_j, of undominated states of node j.

Algorithm 2: Label setting dynamic programming algorithm

---

1 $\Lambda_i = \emptyset$;

2 $\Lambda_o = \{\mathcal{S}((o))\}$;

3 $\Delta_i = \emptyset$

4 $L = L^{init}$

5 while $\cup_{i \in \mathcal{V}} (\Lambda_i \setminus \Delta_i) \neq \emptyset$ **do**

6   **foreach** $i \in \mathcal{V}$ *in topological order* **do**

7 $\Theta_i = \Lambda_i \setminus \Delta_i$

8   **while** $\ell \leq L$ **do**

9 $\mathcal{S}(\mathcal{P}) = \underset{\mathcal{S}(\mathcal{P}') \in \Theta_i}{\arg\min} \left\{ C(\mathcal{P}') \right\}$

10   **foreach** $j \in Succ(i)$ **do**

11   **if** *extendable($\mathcal{S}(\mathcal{P})$, j)* **then**

12 $\mathcal{S}(\mathcal{Q}) = \text{extend}(\mathcal{S}(\mathcal{P}), j)$;

13   **if** *not dominated($\Lambda_j, \mathcal{S}(\mathcal{Q})$)* **then**

14   eliminate_ineffecient($\Lambda_j, \mathcal{S}(\mathcal{Q})$);

15   insert($\Lambda_j, \mathcal{S}(\mathcal{Q})$);

16   **end**

17   **end**

18   **end**

19 $\Delta_i = \Delta_i \cup \{\mathcal{S}(\mathcal{P})\}$

20 $\Theta_i = \Theta_i \setminus \{\mathcal{S}(\mathcal{P})\}$

21 $\ell = \ell + 1$

22   **end**

23   **end**

24 $L = 2L$

25 end

In some cases we can exploit having an upper bound value of the optimal objective value – see Section 4.1. However, Algorithm 1 applies a breadth-first strategy for the extension, and as a consequence the upper bound will decrease slowly throughout the process.

To remedy this we apply a limited extension strategy inspired by Burke and Curtois (2014), who apply dynamic programming for identifying individual schedules in nurse rostering. The intuition is for each node only to extend the L best non-extended states. In our context, the best are those having the lowest objective value. Thus, we divide the execution of the algorithm into a number of stages where – in each stage – we extend (at most) L states from each node in topological order. If any nodes have unextended states after a stage, then we start a new stage, potentially with a larger value of L. By doing this we emphasize a greedy approach in the first stages of the algorithm such that we can obtain a reasonable upper bound solution value, and then, in the later stages, we explore the remaining states.

In Algorithm 2 we describe dynamic programming with limited extension. This is an adaption of Algorithm 1, where we keep track of which states have been extended and split the solution process into stages, where in each stage at most $L|\mathcal{V}|$ states are extended. In addition to Algorithm 1, the set Δ_i stores the states that have already been extended, while the set Θ_i corresponds to the set of non-extended states.

We initialize the algorithm with $L = L^{init}$, where $L^{init} \geq 1$ is the maximum number of states we wish

to extend from each node in the first stage. After each stage we double the number of states, L, we allow to be extended. In line 9 the states are selected in increasing order of cost, such that we greedily select the most promising states in the beginning of the algorithm.[2] If we put $L^{init} = \infty$, then Algorithm 2 will be equivalent to Algorithm 1.

The computational complexity depends on the potential number of states that have to be extended from each node. In the case where the revenue, the mean, and the variance are all integer and we have a strict upper bound in the form of constraint (2), it is possible to (pseudo-) polynomially bound the number of states that are undominated for a given node, see, for instance Desrochers and Soumis (1988). However, in our case we do not necessarily have all integer values for the revenue, mean, or variance, and we do not require (2) to be present. As a consequence, the algorithms are in principle bounded only by the number of potential paths in the graph, which in general is not polynomially bounded. Thus, we are interested in further reducing the number of potential states that do not lead to an optimal solution. We do this in the following section.

4.1 State bounding for convex expected overfill cost functions

Until now we have assumed that f was increasing. If, however, f is also convex, then we can accelerate the solution process by bounding the objective of possible extensions. This bounding relies on a combination of a deterministic knapsack lower bound and the lower bound on the expected overtime given by Proposition 3.

We let UB be the value of the best known solution at any point of the algorithm. Initially, this is set to 0, as it is the trivial solution including no items. Given a path, $\mathcal{P}$, with cost $C(\mathcal{P})$ we let $LB(\mathcal{P})$ be a lower bound on the additional cost of any possible extension of path $\mathcal{P}$ to the destination node, d. With this bound we can eliminate the state $\mathcal{S}(\mathcal{P})$ if

$$C(\mathcal{P}) + LB(\mathcal{P}) \geq UB \tag{15}$$

However, the crux is to identify the lower bound, $LB(\mathcal{P})$. Assume that we have the path $\mathcal{P} = (o, v_1, \ldots, v_p)$ and some feasible extension $\mathcal{Q} = (w_1, \ldots, w_q, d)$ of path $\mathcal{P}$. Then we can construct the full feasible path $(\mathcal{P}, \mathcal{Q}) = (0, v_1, \ldots, v_p, w_1, \ldots, w_q, d)$ as the concatenation of $\mathcal{Q}$ to $\mathcal{P}$. By the cost recursion (14) we must have

$$C(\mathcal{P}, \mathcal{Q}) - C(\mathcal{P}) = -\sum_{j \in \mathcal{Q}} r_j + f\left(h_T\left(\mu(\mathcal{P}, \mathcal{Q}), \sqrt{\sigma^2(\mathcal{P}, \mathcal{Q})}\right)\right) - f\left(h_T\left(\mu(\mathcal{P}), \sqrt{\sigma^2(\mathcal{P})}\right)\right) \tag{16}$$

and we have to identify the lower bound in the cost of any extension, $\mathcal{Q}$, such that $C(\mathcal{P}, \mathcal{Q}) - C(\mathcal{P}) \geq LB(\mathcal{P})$. We do this by observing that the last term of the right-hand side of (16) is constant for any extension, and then construct convex lower bounding functions for the two remaining terms.

Before setting up the lower bound we sort the nodes such that the sequence of the $\mathcal{J}_k$ sets reflects a decreasing potential of improving the objective value. Within each set, $\mathcal{J}_k$, we denote the maximum mean value

$$M_k = \max\left\{\mu_j \,|\, j \in \mathcal{J}_k\right\}, \quad k \in \mathcal{K}$$

---

[2]It should be noted that we have chosen "most promising" to correspond to the cost, but other alternatives could be used, for example, the cost per unit of the mean, $c(\mathcal{P})/\mu(\mathcal{P})$.

Furthermore, we will let the lowest cost per unit of the mean

$$L_k = \min\left\{ \frac{-r_j}{\mu_j} \,\middle|\, j \in \mathcal{J}_k \right\}, \quad k \in \mathcal{K}$$

The value $M_k L_k$ represents a lower bound on the value that we can attain from selecting an element, $j \in \mathcal{J}_k$. That is, $M_k L_k \leq -r_j$ for all $j \in \mathcal{J}_k$. We will assume that set $\mathcal{K}$ is sorted in increasing order of $M_k L_k$ (i.e., for $h, k \in \mathcal{K}$ with $h < k$ then $M_h L_h \leq M_k L_k$).

Now we turn to identifying a lower bound for $\sum_{j \in \mathcal{Q}} -r_j$ for any extension, $\mathcal{Q}$, of path $\mathcal{P}$. This bound is closely related to a lower bound obtained by solving a fractional KP. The lower bound we are searching for is for all possible extensions of a given path, and consequently we are only interested in the subsets, $\mathcal{J}_k$, to which we can extend this path. Hence, for the path $\mathcal{P} = (o, v_1, \ldots, v_p)$ we denote the set

$$\mathcal{E}(\mathcal{P}) = \{k \in \mathcal{K} | k > \max\{h \in \mathcal{K} | v_p \in \mathcal{J}_h\}\} \subseteq \mathcal{K}$$

as the index set of subsets to which path $\mathcal{P}$ can be extended. This corresponds to all the indices of subsets with a larger index than the last subset added.

Suppose that $m \in \mathbb{R}_+$ is a non-negative amount (corresponding to additional mean) that can be spent. Then we can include elements

$$k(\mathcal{P}, m) = \max\left\{ k \in \mathcal{E}(\mathcal{P}) \,\middle|\, \sum_{i \in \mathcal{E}(\mathcal{P}), i \leq k} M_i \leq m \right\}$$

without surpassing the value, m. The next element, $k(\mathcal{P}, m) + 1$, will either not be added or only be added at a fractional value. Due to the fact that $L_k M_k \leq -r_j$ for all $j \in \mathcal{J}_k$ we have the following lower bound on the value of extension with mean m:

$$z(\mathcal{P}, m) = \sum_{\substack{i \in \mathcal{E}(\mathcal{P}) \\ i \leq k(\mathcal{P}, m)}} M_i L_i + L_{k(\mathcal{P}, m)+1} \left(m - \sum_{\substack{i \in \mathcal{E}(\mathcal{P}) \\ i \leq k(\mathcal{P}, m)}} M_i \right) \tag{17}$$

Note that if $\mathcal{J}_k$ contains only one element for each k, then this lower bound corresponds to the LP-relaxed value of a binary knapsack problem with elements from $\mathcal{E}(\mathcal{P})$ and capacity m. As the elements in $\mathcal{K}$ are sorted in increasing order of $M_k L_k$, we have that the slope (but not the function itself) of the function $z(\mathcal{P}, m)$ is non-decreasing in m and it is therefore (weakly) convex. Thus, we have a decreasing convex function as a lower bound on the cost when increasing the mean value.

We now turn our attention to the bounding of $f(h_T(\mu(\mathcal{P}, \mathcal{Q}), \sqrt{\sigma^2(\mathcal{P}, \mathcal{Q})}))$ in (16). We observe from Proposition 3 that f is increasing and that

$$
\begin{aligned}
f\left(h_T\left(\mu(\mathcal{P}, \mathcal{Q}), \sqrt{\sigma^2(\mathcal{P}, \mathcal{Q})} \right) \right) &\geq f\left(\max\{0; \mu(\mathcal{P}, \mathcal{Q}) - T\} \right) \\
&= f\left(\max\left\{ 0; \mu(\mathcal{P}) + \sum_{j \in \mathcal{Q}} \mu_j - T \right\} \right) \\
&= f\left(\max\{0; \mu(\mathcal{P}) + m - T\} \right)
\end{aligned}
$$

where $m = \sum_{j \in \mathcal{Q}} \mu_j$. Then note that $\mu(\mathcal{P})$ and T are constants and that $\max\{\mu(\mathcal{P}) + m - T\}$ is a convex function in m. As a consequence we have that $f(\max\{0; \mu(\mathcal{P}) + m - T\})$ is a convex function in m.

We have now expressed the lower bound $z(\mathcal{P}, m)$ for $\sum_{j \in \mathcal{Q}} -r_j$ when $\mathcal{Q}$ is allowed to use at most m in mean, and likewise we have the lower bound $f(\max\{0; \mu(\mathcal{P}) + m - T\})$ for $f(h_T(\mu(\mathcal{P}, \mathcal{Q}), \sqrt{\sigma^2(\mathcal{P}, \mathcal{Q})}))$ when $\mathcal{Q}$ is allowed to use m as mean. Both of these lower bounds are convex functions and so will the sum of these be. Consequently, minimizing the sum of the lower bound will yield a unique minimum (if one exists) when we minimize over m. The additional mean, m, is at least 0. It is, however, bounded from above by (2), where we let $\beta = 0$ (i.e., $m \leq T + S - \mu(\mathcal{P})$). Thus, the domain we are minimizing over is $0 \leq m \leq T + S - \mu(\mathcal{P})$. We denote the domain $\mathcal{M}(\mathcal{P}) = [0; T + S - \mu(\mathcal{P})]$, and state the lower bound on the cost of any extension as

$$LB(\mathcal{P}) = \min_{m \in \mathcal{M}(\mathcal{P})} \left\{ z(\mathcal{P}, m) + f\left(\max\{0; \mu(\mathcal{P}) + m - T\}\right) \right\} - f\left(h_T\left(\mu(\mathcal{P}), \sqrt{\sigma^2(\mathcal{P})}\right)\right) \qquad (18)$$

Note that the convexity of f is only used to guarantee that a local optimum is a global optimum in (18) when minimizing. By combining (15) with (18), we obtain the following proposition.

Proposition 5. *Let $\mathcal{P}$ be a feasible path from o to j and let UB be an upper bound on the solution value. If*

$$\min_{m \in \mathcal{M}(\mathcal{P})} \left\{ z(\mathcal{P}, m) + f\left(\max\{0; \mu(\mathcal{P}) + m - T\}\right) \right\} \geq UB + \sum_{i \in \mathcal{P}} r_i \qquad (19)$$

then no extension of $\mathcal{P}$ will attain a lower objective value than UB.

The constraint (15) applying the bound is in our case used on time of creation of the state and on time of extension of the state. This corresponds to line 7 and line 6, respectively, in Algorithm 1 and line 12 and line 11, respectively, in Algorithm 2. In practice, we calculate the left-hand side of (19) when creating the state and then we check and use this cached value whenever doing the actual check.

4.1.1 Numerical example

In this section we will provide a small numerical example of the lower bound for a specific convex cost function. Suppose that $f(x) = \frac{1}{2}x^2$, which is non-decreasing and convex in the interval of non-negative numbers. Let $T = 100$ and $S = 20$. Let $\mathcal{P}$ be a path that has resulted in a mean of $\mu(\mathcal{P}) = 94$. The interval of the additional mean is then $\mathcal{M}(\mathcal{P}) = [0; 100 + 20 - 94] = [0; 26]$. For simplicity we assume that the items are not mutually excluding, meaning that the sets, $\mathcal{J}_k$, are all singleton sets, and we put $\mathcal{J}_k = \{k\}$. Suppose that we are given four items, which have not yet been used, with mean and revenue given in Table 1. The elements are sorted in increasing order of L_k and in this case there is one-to-one

$j(=k)$	r_j	μ_j	M_k	L_k
a_1	10	5	5	-2
a_2	8	5	5	$-1\frac{3}{5}$
a_3	15	10	10	$-1\frac{1}{2}$
a_4	10	10	10	-1

Table 1: Data for example of a lower bound

correspondence between the elements of $\mathcal{J}$ and the elements of the set $\mathcal{K}$. Furthermore, these items

16

constitute the set $\mathcal{E}(\mathcal{P}) = \{a_1, \ldots, a_4\}$.

Now we need to identify $m \in \mathcal{M}(\mathcal{P})$, which minimized $z(\mathcal{P}, m) + f(\max\{0; \mu(\mathcal{P}) + m - T\})$ on the left-hand side of (19). We note for $m \leq T - \mu(\mathcal{P}) = 100 - 94 = 6$ that $\max\{0; \mu(\mathcal{P}) + m - T\} = 0$ and as the slope is negative for $z(\mathcal{P}, m)$ the optimal value of m is trivially 6 in the interval $[0; 6]$. For $m > 6$ it is more interesting. We observe that $z(\mathcal{P}, m)$ changes slope, given by L_k, in the points 5, 6, 10, and 20. In this example we denote $g(m) = z(\mathcal{P}, m) + f(\max\{0; \mu(\mathcal{P}) + m - T\})$, which can be stated explicitly on the domain $[0; 26]$ as

$$
g(m) = \begin{cases}
 & -2m, & 0 \leq m < 5 \\
 & -1\frac{3}{5}(m-5) \quad -10, & 5 \leq m < 6 \\
\frac{1}{2}(m-6)^2 & -1\frac{3}{5}(m-5) \quad -10, & 6 \leq m < 10 \\
\frac{1}{2}(m-6)^2 & -1\frac{1}{2}(m-10) \quad -18, & 10 \leq m < 20 \\
\frac{1}{2}(m-6)^2 & -1(m-20) \quad -33, & 20 \leq m \leq 26
\end{cases}
$$

We have illustrated part of the function $g(m)$ in Figure 3. The optimal value is found for $m = 7\frac{3}{5}$ yielding a value of $-12\frac{22}{25}$.
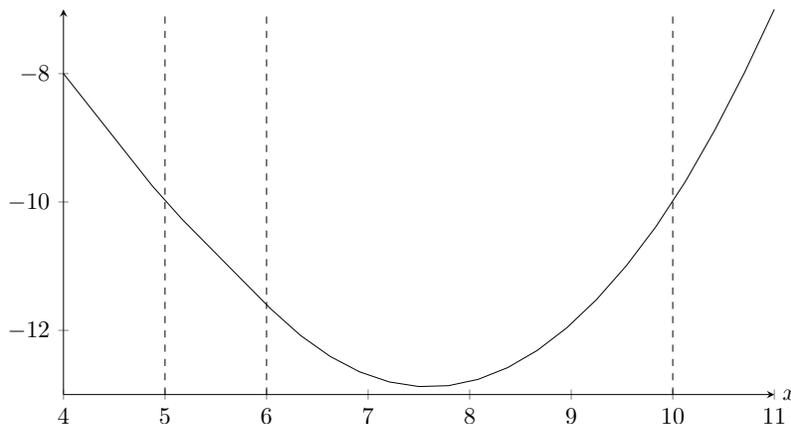

Figure 3: Illustration of the function $g(m)$.

4.1.2 A note on complexity

Finding the minimum of this function amounts to identifying the minimum of each of the individual functions. If for any of the individual functions we obtain a minimum outside the domain for which it contributes to $g(m)$, then we can, due to convexity, disregard all the individual functions in the opposite direction. That is, if the minimum falls below the boundary, then we can disregard all functions with larger boundaries than the current, and likewise if the minimum falls above the boundary, then we can disregard all functions with smaller boundaries than the current function. Thus, in the worst case we have to find the minimum of $O(\log J)$ functions to identify the minimum of $g(m)$. As we can explicitly calculate the minimum of the quadratic function in $O(1)$ time complexity, we have that the complexity of calculating this bound will be $O(\log J)$.

17

5 Computational Study

To illustrate the most important properties of the shortest-path-based approach for solving the SKP we conduct a computational study. First, in Section 5.1 we describe the instances we have generated to conduct the computational study. And second, we describe and discuss the computational results in Section 5.2.

5.1 Test instances

In order to test the algorithms we have generated a number of random instances. These instances are based on generating lists of items with randomly generated revenue, mean, and variance, and we combine each of these with scenarios determining parameters such as maximal fill size and expected overfill penalty function.

The instances with sets of items are generated as follows: First we have three categories, small, medium, and large, based on the number of items having 100, 250, and 500 items, respectively. Within each of these categories a further subdivision based on the variance is made. This subdivision is small, medium, and larger variance. Finally, we have two types of groupings of items – one where all items can be used in conjunction and one where the items are in groups of five being mutually exclusive. Hence, we have 18 distinct categories of instances. Within each category 20 random instances are generated, where the revenue, r_j, for each item is drawn from the uniform distribution between 0 and 100, and the expected weight, μ_j, for each item is drawn from the uniform distribution between 10 and 50. When the variance is small we draw σ_j^2 from the uniform distribution between 0 and 10, whereas for medium and large variances we draw from the uniform distributions between 10 and 25, and 25 and 50, respectively. In all we have 360 distinct item sets. A summary of the setup of the item sets is given in Table 2.

Parameter	Type	Value
J	Small	100
	Medium	250
	Large	500
r_j	-	$U(0, 100)$
μ_j	-	$U(10, 50)$
σ^2	Small	$U(0, 10)$
	Medium	$U(10, 25)$
	Large	$U(25, 50)$
$\mathcal{K}$	No groups	$|\mathcal{K}| = |\mathcal{J}|$
	Groups of 5	$|\mathcal{J}_k| = 5,\ k \in \mathcal{K},\ \mathcal{K} = \{1, \ldots, |\mathcal{J}|/5\}$

Table 2: Setup of the item sets.

For each item set we can test a number of scenarios based on other parameters of the KPs. These parameters are chosen in the following way. The limit of the total fill, T, is tested for small, medium, and large values, being 100, 250, and 500, respectively. Recall that we may violate this limit by additional S units. We test this for S being 0, corresponding to the case where a violation of the T units is prohibited, $T/5$ (i.e., a proportion of T), and infinite, stating that no upper limit on the violation exists. β states the number of standard deviations with which we should be within the limit of $T + S$. We test the cases where

β is equal to 0, 1, and 2. Finally we can test two different types of expected overfill penalty functions. In the first case we assume the function to be a (specific) linear increasing function $f(x) = 5x$, whereas in the second case we assume the function to be a (specific) quadratic function, $f(x) = \frac{1}{2}x^2$, of the expected overfill. In all we have 54 distinct scenarios, and a summary of the scenarios is given in Table 3. If we

Parameter	Type	Value
T	Small	100
	Medium	250
	Large	500
S	T is the limit	0
	Proportion	$T/5$
	No upper limit	∞
β	bounded by mean	0
	bounded by mean plus 1 std. dev.	1
	bounded by mean plus 2 std. dev.	2
$f(x)$	Linear	$5x$
	Quadratic	$\frac{1}{2}x^2$

Table 3: Different scenarios used for each random instance.

combine the generated sets of items with the scenarios we have 19,440 test cases in all.

5.2   Computational results

We have conducted a series of tests based on the test cases described in Section 5.1. The algorithms were implemented in C++ using the TDM-GCC 5.1.0 64bit g++ compiler with the option -O3 turned on. The experiments were conducted on a system equipped with a Xeon CPU EX-2630 v4 @2.20GHz processor having 10 physical cores and 32Gb RAM. We executed nine instances concurrently.[3] The computer had a Windows 7 operating system. Each execution of a test instance was given a maximum of 300 seconds. If the time limit was reached, then the test was prematurely terminated and the best upper bound was reported.

We compare four different approaches based on the shortest path formulation of the problem. These are:

FNb: Using full (F) extension of each node with no bounding (Nb). This corresponds to having $L^{init} = \infty$ and $LB(\mathcal{P}) = -\infty$.

INb: Using incremental (I) extension of each node with no bounding (Nb). This corresponds to having $L^{init} = 1$ and $LB(\mathcal{P}) = -\infty$.

FB: Using full (F) extension of each node with bounding (B). This corresponds to having $L^{init} = \infty$ and using (19) for bounding.

IB: Using incremental (I) extension of each node with bounding (B). This corresponds to having $L^{init} = 1$ and using (19) for bounding.

---

[3]The concurrent execution gives an overhead compared to solving the problems using a single thread. However, we choose to do this to accelerate the overall testing process.

		FNb			INb			FB			IB		
$|J|$	T	S	U	D	S	U	D	S	U	D	S	U	D
100	100	70.32	29.68	0.00	71.20	28.80	0.00	100.00	0.00	0.00	100.00	0.00	0.00
100	250	43.29	27.03	29.68	30.60	40.60	28.80	100.00	0.00	0.00	100.00	0.00	0.00
100	500	11.62	31.67	56.71	3.33	27.27	69.40	94.03	5.97	0.00	100.00	0.00	0.00
250	100	65.28	34.72	0.00	62.13	37.87	0.00	100.00	0.00	0.00	100.00	0.00	0.00
250	250	0.00	65.28	34.72	0.00	62.13	37.87	83.47	16.53	0.00	99.81	0.19	0.00
250	500	0.00	0.00	100.00	0.00	0.00	100.00	10.37	73.10	16.53	89.30	10.51	0.19
500	100	40.74	59.26	0.00	22.87	77.13	0.00	99.03	0.97	0.00	99.86	0.14	0.00
500	250	0.00	40.74	59.26	0.00	22.87	77.13	48.19	50.84	0.97	93.80	6.06	0.14
500	500	0.00	0.00	100.00	0.00	0.00	100.00	0.00	48.19	51.81	60.70	33.10	6.20

Table 4: Pct. instances solved within the time limit. For each combination of $|J|$ and T 2160 instances-scenario combinations are possible. For each algorithm, FNb, INb, FB, and IB, column S gives the pct. solved, column U gives the pct. of tested but unsolved instances, and column D indicates the pct. of disregarded instances. An instance-scenario combination is disregarded if the same algorithm has failed to solve a corresponding instance-scenario with the only difference that it has a smaller T value.

All of these are tested using Algorithm 2, where we note that when $L^{init} = \infty$, then Algorithm 2 is equivalent to Algorithm 1.

Table 4 shows for each of the variants of the algorithms the percentage of instances solved within the time limit of 300 seconds. Each row in the table corresponds to a combination of the number of items $|J|$ and the limit, T, as these are the primary indicators of how difficult an instance is to solve. For each combination of number of items $|J|$ and fill limit, T, we have 2160 instance-scenario combinations. The first two columns indicate $|J|$ and T. The remaining columns are divided into four sections – one for each approach tested. For each approach three columns are given. The first column (S) indicates how large a percentage of the 2160 instances the algorithm solved to optimality. The second column (U) indicates the percentage of the 2160 instances that were tested but unsolved due to the time limit. The last column (D) indicates the percentage of instances that were disregarded in the testing. We have disregarded instances for which the algorithm has reached the time limit for a lower value of T, as they would (most likely) reach the time limit too.

It is not surprising that when we increase either $|J|$ or T, then any of the algorithms will time out on more instances (except in the case where all instances are solved to optimality). In the case where we use no bound (FNb and INb) we observe that the full extension approach (FNb) solves more instances than the incremental method (INb). This is due to a slight increase in the number of dominance checks, which is necessary when we use the incremental approach compared to the full approach. We see quite the opposite picture when comparing the full extension approach using bounding (FB) with the incremental extension approach using bounding (IB). In this case IB solves significantly more instances than FB. The reason is that IB obtains a tighter upper bound much faster than FB and can, as a consequence, eliminate states earlier in the process. Furthermore, IB is indeed the only of the tested approaches that is able to solve any of the instances with $|J| = 500$ and $T = 500$ within the time limit. We observe that using the bound significantly increases the number of instances that can be solved by the algorithms.

Figure 4 shows to the left the cumulative number of instances each method is able to solve within a given time and to the right the cumulative number of instances for which each method reaches the best solution no later than a given time. We observe that the IB method solves more instances within two-and-a-half seconds than any of the other methods do in 300 seconds.[4] Similarly, we see that FB

---

[4]The IB method has solved 13,900 instances within 2.5 sec., while the FNb, the INb, and the FB have solved 5014
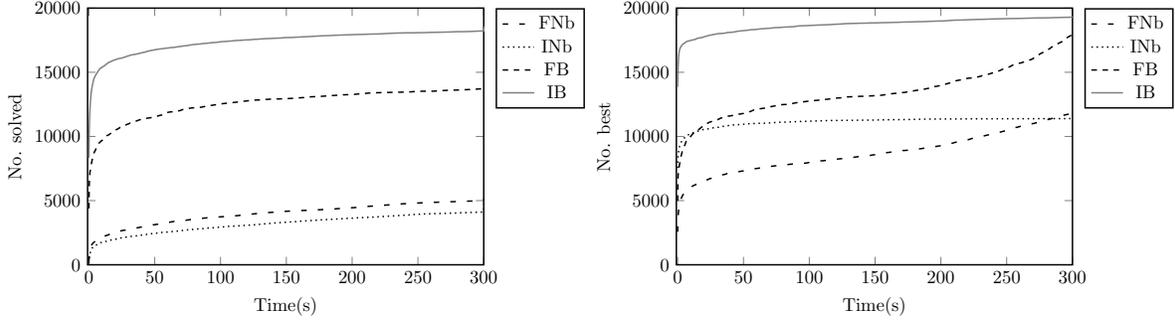
Figure 4: The horizontal axis is time in seconds, while the vertical axis is the accumulated number of instances. Left: The cumulative number of instances solved within time. Right: The cumulative number of instances, where the best solution is found within time.

solves significantly more instances (7367 instances to be precise) within the first second than the two approaches – FNb and INb – not using the bound within 300 seconds. Hence, using the bound-based elimination from Proposition 5 is superior to not using this elimination.

The cumulative number of instances that obtain the best solution within a given time can be seen in the right panel of Figure 4. That is, after this time no better solution is found. In many cases the best solution is found very early. However, the behavior of the full extension methods (FNb and FB) is different from that of the incremental methods (INb and IB). While the full extension methods find new best solutions throughout the 300 seconds, the incremental methods find most of the best solutions very quickly. Moreover, these solutions are often optimal. Consequently, the incremental approach yields a reasonable heuristic for the SKP.

In general, we observe that for the approaches not using the bound (FNb and INb) it is easier to solve the grouped instances. This is due to the number of potential paths in the graph becoming significantly smaller when some of the items are mutually exclusive. However, we see the opposite effect for the approaches using the bound (FB and IB). This is due to a weakening of the bounding when we group the items, and therefore we cannot eliminate as many states based on the bound.

The IB approach is superior to the other approaches and we therefore turn to this approach and investigate which cases are difficult for this approach to solve. In order to do this we limit our attention to the set of instances having $|\mathcal{J}| = 500$ and $T = 500$, as we anticipate that the effects will be most pronounced in these instances.

In Figure 5 we illustrate how many instances are solved within the time limit when we vary β, S, and the variance of the item size. To the left we observe that when $S = 0$, the number of instances solved decreases drastically when we increase β. The reason for this is that the number of states eliminated due to the lower bound from Proposition 5 decreases significantly. The lower bound (18) does not incorporate the maximal overfill constraint (2), while the upper bound satisfies this constraint. As a consequence, the gap between these two bounds becomes too large for constraint (19) to be satisfied. We do not see the same effect for $S = T/5$ or for $S = \infty$, as (2) is less constraining when identifying the upper bound.

To the right in Figure 5 we see that it is easier to solve instances with small variance than instances with large variance. Again, the lower bound calculation in (18) becomes weaker when the variance

---

instances, 4258 instances, and 13725 instances, respectively, within 300 sec.
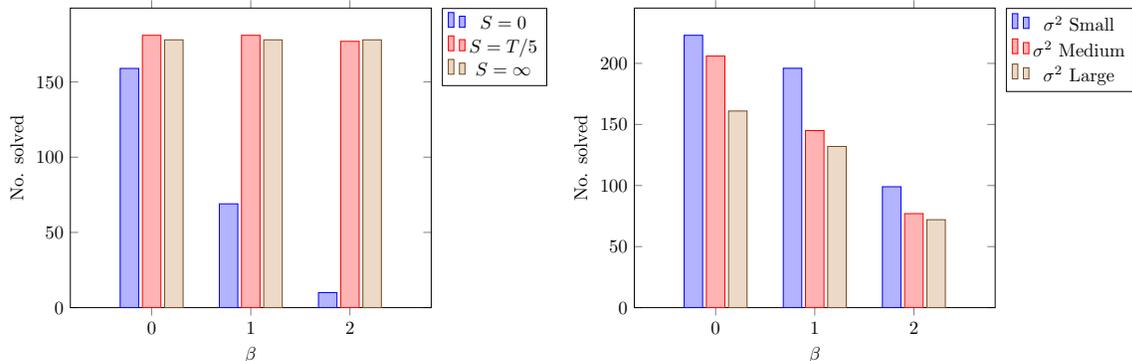
Figure 5: The number of instances solved for different values of $\beta \in \{0, 1, 2\}$ by IB when $|J| = 500$ and $T = 500$. Left: The number of instances solved for combinations of β and S; $S = 0$, $S = T/5$, and $S = \infty$. Right: The number of instances solved for combinations of β and variance of the instances; Small ($\sigma^2 \in [0, 10]$), Medium ($\sigma^2 \in [10, 25]$), and Large ($\sigma^2 \in [25, 50]$).

increases, as we have applied Proposition 3 in order to obtain a deterministic lower bound for the penalty of the expected overfill, and we know that the expected overfill increases with the variance, as illustrated in Figure 3 and proved in Proposition 2.
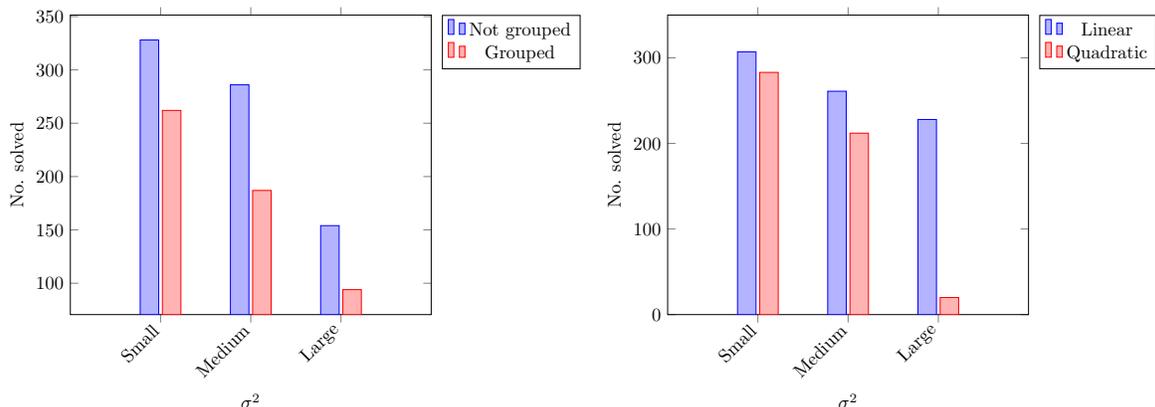

Figure 6: The number of instances solved by IB when $|J| = 500$ and $T = 500$ for different variances; Small ($\sigma^2 \in [0, 10]$), Medium ($\sigma^2 \in [10, 25]$), and Large ($\sigma^2 \in [25, 50]$). Left: The number of instances solved based on grouping of items; Not grouped ($|\mathcal{K}| = |\mathcal{J}|$) and Grouped ($|\mathcal{J}_k| = 5$ for all $k \in \mathcal{K}$ and $\mathcal{K} = \{1, \ldots, |\mathcal{J}/5|\}$). Right: The number of instances solved when the penalty function is linear, $f(x) = 5x$, and quadratic, $f(x) = \frac{1}{2}x^2$.

To further illustrate the effects we show in Figure 6 the number of instances the IB algorithm is able to solve for different levels of variance and for different groupings and penalty functions. As observed above, more instances can be solved when the variance is small compared to when it is large. It is also observed that it is easier to solve the non-grouped instances compared to the grouped instances for the IB algorithm. And finally, it is easier to solve problems when the penalty function is linear than when it is quadratic. In fact, when the instances are grouped, the variance is larger, and we use the quadratic penalty function, then none of the instances of size $\mathcal{J} = 500$ and $T = 500$ are solved to optimality by the IB algorithm. Again, this is due to the gap between the upper and lower bounds becoming too large for

constraint (19) to be satisfied, and therefore fewer states are eliminated early in the algorithm.

To summarize the computational study, we observe that combining incremental extensions of states with the bounding procedure yields by far the best results. It is observed that the incremental bounding needs to be supplemented by the use of a bounding procedure to outperform the full extension approach. Furthermore, we observe that the bounding is effective in many cases, but is challenged in the cases where variance is large. Finally, but not surprisingly, when the gap between the upper bound solution and the lower bound used becomes large, then the IB approach will have difficulty solving the instances. Thus, to improve the method the incorporation of constraint (2) could be investigated, but we have left this for future research.

6   Concluding remarks and future research

In this paper we have illustrated how a stochastic knapsack problem (SKP) can be modeled as a shortest path problem with resource constraints (SPPRC). We have demonstrated that applying incremental extension combined with a state bounding procedure yields the best results of the approaches tested. Furthermore, it is shown that the incremental extension approach provides a reasonable heuristic for the problem.

In relation to our motivation for solving the problem, we have applied this approach to a generalized assignment problem based formulation for the dynamic surgery allocation problem of Range et al. (2016). In this context the number of potential surgeries within the T time units is not larger, and our variant of the SKP works very well as the pricing problem of a column generation procedure.

We have limited ourselves to the case where we have sets of mutually excluding items as well as convex cost of overfilling and bounds on the potential overfilling. The method is not limited to this, however. Below we give examples of potential extensions of the model, which can easily be added.

As the problem is formulated as an SPPRC, it is possible to include more resource constraints. For instance, if a bound on the maximal number of items is present, then it can be formulated by using a resource extension function and resource windows on each of the node in the item graph. Likewise, it is possible to include a resource extension function and corresponding resource windows guaranteeing that at least a specific number of items are selected.

In the case where we have an integer knapsack problem, each of the item types can be formulated as a group, where visiting a node in the group corresponds to selecting a specific number of that item type. In this case, it is possible to have individual revenues – which are not necessarily linear in the number of items – on the number of items selected of each type. This could be the case when it is desirable to have multiple items of the same type in the solution.

Fundamentally, knapsack problems do not incorporate sequencing of the items selected to be in the knapsack. However, the shortest-path-based approach described in this paper can be generalized to include sequencing. This would require that we reformulate the acyclic graph to a potentially cyclic graph where we would only be allowed to visit each group maximally once. As a consequence, the problem becomes an elementary shortest path problem with resource constraints. This problem is, however, strongly $\mathcal{NP}$-hard, and one should expect that the size of the instances that we would be able to solve is significantly smaller than the size of the instances discussed in the present paper.

References

Burke, E. K. and Curtois, T. (2014). New approaches to nurse rostering benchmark instances. *European Journal of Operational Research*, 237(1):71 – 81.

Carraway, R. L., Schmidt, R. L., and Weatherford, L. R. (1993). An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns. *Naval Research Logistics*, 40(2):161–173.

Chabrier, A. (2006). Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972 – 2990. Part Special Issue: Constraint Programming.

Chen, K. and Ross, S. M. (2015). Static stochastic knapsack problems. *Prob. Eng. Inf. Sci.*, 29(04):527–546.

Dean, B. C., Goemans, M. X., and Vondrák, J. (2008). Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of OR*, 33(4):945 – 964.

Derman, C., Lieberman, G. J., and Ross, S. M. (1978). A renewal decision problem. *Management Science*, 24(5):554–561.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M. M., Soumis, F., and Villneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In Crainic, T. G. and Laporte, G., editors, *Fleet Management and Logistics*, pages 57–93. Kluwer.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354.

Desrochers, M. and Soumis, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, 26(3):191–212.

Eveborn, P. and Rönnqvist, M. (2004). Scheduler – a system for staff planning. *Annals of Operations Research*, 128(1):21–45.

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229.

Gauvin, C., Desaulniers, G., and Gendreau, M. (2014). A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 50:141 – 153.

Henig, M. I. (1990). Risk criteria in a stochastic knapsack problem. *Operations Research*, 38(5):820–825.

İlhan, T., Iravani, S. M. R., and Daskin, M. S. (2011). The adaptive knapsack problem with stochastic rewards. *Operations Research*, 59(1):242 – 248.

Irnich, S. (2008). Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers, G., Desrosiers, J., and Solomon, M., editors, *Column Generation*, pages 33–65. Springer US.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kleywegt, A. J. and Papastavrou, J. D. (2001). The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, 49(1):26–41.

Kleywegt, A. J., Shapiro, A., and de Mello, T. H. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502.

Kosuch, S. and Lisser, A. (2009). Upper bounds for the 0-1 stochastic knapsack problem and a b&b algorithm. *Ann Oper Res*, 176(1):77–93.

Lozano, L. and Medaglia, A. L. (2013). On an exact method for the constrained shortest path problem. *Computers & Operations Research*, 40(1):378 – 384.

Lusby, R., Dohn, A., Range, T. M., and Larsen, J. (2012). A column generation-based heuristic for rostering with work patterns. *Journal of the Operational Research Society*, 63(2):261–277.

Merzifonluoğlu, Y., Geunes, J., and Romeijn, H. (2012). The static stochastic knapsack problem with normally distributed item sizes. *Mathematical Programming*, 134(2):459–489.

Morton, D. P. and Wood, R. K. (1998). On a stochastic knapsack problem and generalizations. In Woodruff, D. L., editor, *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research*, pages 149–168. Springer US, Boston, MA.

Range, T. M., Kozlowski, D., and Petersen, N. C. (2016). Dynamic job assignment: A column generation approach with an application to surgery allocation. Discussion Papers on Business and Economics, University of Southern Denmark.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255 − 273. Graphs and Combinatorial Optimization.

Smet, P., Ernst, A. T., and Berghe, G. V. (2016). Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem. *Computers & Operations Research*, 76:60 − 72.

Sniedovich, M. (1980). Preference order stochastic knapsack problems: Methodological issues. *The Journal of the Operational Research Society*, 31(11):1025 − 1032.

Steinberg, E. and Parks, M. S. (1979). A preference order dynamic program for a knapsack problem with stochastic rewards. *The Journal of the Operational Research Society*, 30(2):141–147.

Tönissen, D., van den Akker, J., and Hoogeveen, J. (2017). Column generation strategies and decomposition approaches for the two-stage stochastic multiple knapsack problem. *Computers & Operations Research*, 83:125 − 139.

A   Proofs

Proof of Proposition 1. Assume that $X_1, \ldots, X_n$ are independent stochastic variables with means μ_i and variances σ_i^2 for $i = 1, \ldots, n$, and that $T \geq 0$ is a constant. Denote $X = X_1 + \ldots + X_n$, $\mu_X = \mu_1 + \ldots + \mu_n$ and $\sigma_X^2 = \sigma_1^2 + \ldots + \sigma_n^2$. By the central limit theorem we have that X is approximately normally distributed (i.e., $X \sim \mathcal{N}\left(\mu_X, \sigma_X^2\right)$).

We can now calculate the expected value of the stochastic variable $O = (X - T)^+$

$$\mathbb{E}[O] \approx \int_T^\infty (x - T) f_X(x) dx \tag{20}$$

Let

$$T = \mu_X + k\sigma_X \tag{21}$$

Then we have

$$\mathbb{E}[O] \approx \int_{\mu_X + k\sigma_X}^\infty (x - \mu_X - k\sigma_X) f_X(x) dx \tag{22}$$

$$= \int_{\mu_X + k\sigma_X}^\infty (x - \mu_X - k\sigma_X) \frac{1}{\sigma_X \sqrt{2\pi}} e^{-\frac{(x - \mu_X)^2}{2\sigma_X^2}} dx \tag{23}$$

After substituting $u = (x - \mu_X)\sigma_X$ and simplifying we get

$$\mathbb{E}[O] \approx \sigma_X \int_k^\infty (u - k) \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du, \tag{24}$$

which is a special function of the unit normal distribution. Specifically, let

$$G_u(k) = \int_k^\infty (u - k) \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du \tag{25}$$

Using the special property of the unit normal distribution that

$$\int_k^\infty u \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du = \frac{1}{\sqrt{2\pi}} e^{-\frac{k^2}{2}} \tag{26}$$

(25) can be expressed as

$$G_u(k) = \frac{1}{\sqrt{2\pi}} e^{-\frac{k^2}{2}} - k \int_k^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du \tag{27}$$

$$= \phi(k) - k(1 - \Phi(k)) \tag{28}$$

where ϕ and Φ denote probability density function and cumulative distribution function of a unit normal random variable, respectively. As a result the expected amount of overtime can be represented as a simple function of unit normal distribution

$$\mathbb{E}[O] \approx \sigma_X \left[\phi(k) - k(1 - \Phi(k))\right] \tag{29}$$

where $k = (T - \mu_X)/\sigma_X$. $\square$

Proof of Proposition 2. Let

$$
\begin{aligned}
h_T(\mu,\sigma) &= \sigma\left[\phi\left(\tfrac{T-\mu}{\sigma}\right) - \tfrac{T-\mu}{\sigma}\left(1 - \Phi\left(\tfrac{T-\mu}{\sigma}\right)\right)\right] \\
&= \sigma\phi\left(\tfrac{T-\mu}{\sigma}\right) + (T-\mu)\Phi\left(\tfrac{T-\mu}{\sigma}\right) - (T-\mu)
\end{aligned}
$$

for fixed T. First we identify the first order derivative of $h_T(\mu,\sigma)$ with respect to μ

$$
\begin{aligned}
\frac{\partial h_T(\mu,\sigma)}{\partial \mu} &= \sigma\frac{\partial}{\partial\mu}\left(\phi\left(\tfrac{T-\mu}{\sigma}\right)\right) - \frac{\partial}{\partial\mu}(T-\mu) + \frac{\partial}{\partial\mu}\left((T-\mu)\Phi\left(\tfrac{T-\mu}{\sigma}\right)\right) \\
&= \sigma\tfrac{T-\mu}{\sigma^2}\phi\left(\tfrac{T-\mu}{\sigma}\right) + 1 - \tfrac{T}{\sigma}\phi\left(\tfrac{T-\mu}{\sigma}\right) - \Phi\left(\tfrac{T-\mu}{\sigma}\right) + \tfrac{\mu}{\sigma}\phi\left(\tfrac{T-\mu}{\sigma}\right) \\
&= 1 - \Phi\left(\tfrac{T-\mu}{\sigma}\right)
\end{aligned}
$$

We can calculate the first order derivative of $h_T(\mu,\sigma)$ with respect to σ as

$$
\begin{aligned}
\frac{\partial h_T(\mu,\sigma)}{\partial\sigma} &= \frac{\partial}{\partial\sigma}\left(\sigma\phi\left(\tfrac{T-\mu}{\sigma}\right)\right) + (T-\mu)\frac{\partial}{\partial\sigma}\left(\Phi\left(\tfrac{T-\mu}{\sigma}\right)\right) \\
&= \frac{\partial}{\partial\sigma}\left(\frac{\sigma}{\sqrt{2\pi}}e^{-\left(\frac{T-\mu}{\sigma}\right)^2/2}\right) + (T-\mu)\frac{\partial}{\partial\sigma}\left(\int_{-\infty}^{\frac{T-\mu}{\sigma}}\frac{1}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}du\right) \\
&= \frac{1}{\sqrt{2\pi}}\left(\frac{T-\mu}{\sigma}\right)^2 e^{-\left(\frac{T-\mu}{\sigma}\right)^2/2} + \frac{1}{\sqrt{2\pi}}e^{-\left(\frac{T-\mu}{\sigma}\right)^2/2} - \frac{1}{\sqrt{2\pi}}\left(\frac{T-\mu}{\sigma}\right)^2 e^{-\left(\frac{T-\mu}{\sigma}\right)^2/2} \\
&= \frac{1}{\sqrt{2\pi}}e^{-\left(\frac{T-\mu}{\sigma}\right)^2/2} \\
&= \phi\left(\tfrac{T-\mu}{\sigma}\right)
\end{aligned}
$$

$\square$

Proof of Proposition 3. From Proposition 2 we know that for $\sigma > 0$ the function of expected overfill is strictly increasing, and consequently strictly decreasing when we decrease σ. We now show that when we decrease σ towards zero the expected overtime will converge to $\max\{0; \mu - T\}$.

Suppose that we decrease σ towards zero. Then we observe three cases:

$T - \mu = 0$: Only the term $\sigma\phi\left(\tfrac{T-\mu}{\sigma}\right)$ matters, as the remaining terms are zero. We have that $\phi(0) = 1/\sqrt{2\pi}$ and as a result $\sigma\phi(0) \to 0$ as $\sigma \to 0$.

$T - \mu > 0$: In this case we have that for $\sigma \searrow 0$ then $\tfrac{T-\mu}{\sigma} \nearrow \infty$ and for some number $k \nearrow \infty$ we have that $\phi(k) \to 0$ and $\Phi(k) \to 1$. Thus $h_T(\mu,\sigma) \to 0$ for $\sigma \searrow 0$.

$T - \mu < 0$: When $T - \mu$ is negative we have that $\tfrac{T-\mu}{\sigma} \searrow -\infty$ when $\sigma \searrow 0$ and for some number $k \searrow 0$ we have that $\phi(k) \to 0$ and $\Phi(k) \to 0$, and consequently $h_T(\mu,\sigma) \to -(T-\mu) = \mu - T$.

Thus, the function $h_T(\mu,\sigma)$ converges monotonically from above towards $\max\{0; \mu - T\}$. $\square$

Proof of Proposition 4. The proof is in two parts, where the first part proves that any feasible extension of $\mathcal{P}_2$ is also a feasible extension of $\mathcal{P}_1$ while the second part is to prove that for any extension $\mathcal{Q}$ of $\mathcal{P}_2$ the same extension of $\mathcal{P}_1$ will result in no larger costs (i.e., that $C(\mathcal{P}_1, \mathcal{Q}) \leq C(\mathcal{P}_2, \mathcal{Q})$).

To show that any extension of $\mathcal{P}_2$ is also a feasible extension of $\mathcal{P}_1$ amounts to checking (7) because (6) and (8) are automatically satisfied due to the network structure. Suppose that $\mathcal{Q}$ is a feasible extension

of $\mathcal{P}_2$, then we have that $\mu(\mathcal{P}_2, \mathcal{Q}) + \beta\sqrt{\sigma^2(\mathcal{P}_2, \mathcal{Q})} \leq T + S$. But as both $\mu(\cdot)$ and $\sigma^2(\cdot)$ are additive we have that $\mu(\mathcal{P}_2, \mathcal{Q}) = \mu(\mathcal{P}_2) + \mu(\mathcal{Q})$ and $\sigma^2(\mathcal{P}_2, \mathcal{Q}) = \sigma^2(\mathcal{P}_2) + \sigma^2(\mathcal{Q})$. Consequently, we have

$$
\begin{aligned}
T + S \quad &\geq \quad \mu(\mathcal{P}_2, \mathcal{Q}) + \beta\sqrt{\sigma^2(\mathcal{P}_2, \mathcal{Q})} \\
&= \quad \mu(\mathcal{P}_2) + \mu(\mathcal{Q}) + \beta\sqrt{\sigma^2(\mathcal{P}_2) + \sigma^2(\mathcal{Q})} \\
&\geq \quad \mu(\mathcal{P}_1) + \mu(\mathcal{Q}) + \beta\sqrt{\sigma^2(\mathcal{P}_1) + \sigma^2(\mathcal{Q})} \\
&= \quad \mu(\mathcal{P}_1, \mathcal{Q}) + \beta\sqrt{\sigma^2(\mathcal{P}_1, \mathcal{Q})}
\end{aligned}
$$

showing that $\mathcal{P}_1$ satisfies (7) and therefore any feasible extension of $\mathcal{P}_2$ will also be a feasible extension of $\mathcal{P}_1$.

Now we turn to show that the feasible extension $\mathcal{Q}$ of $\mathcal{P}_2$ will yield no larger cost when applied to $\mathcal{P}_1$ than when it is applied to $\mathcal{P}_2$. First observe that when (1)-(3) of Proposition 4 are satisfied then $-r(\mathcal{P}_1) \leq -r(\mathcal{P}_2)$ and

$$
f\left(h_T\left(\mu(\mathcal{P}_1), \sqrt{\sigma^2(\mathcal{P}_1)}\right)\right) \leq f\left(h_T\left(\mu(\mathcal{P}_2), \sqrt{\sigma^2(\mathcal{P}_2)}\right)\right)
$$

due to Proposition 2. Therefore we know that $C(\mathcal{P}_1) \leq C(\mathcal{P}_2)$. Secondly, due to the additivity of the revenue we will obtain the same difference in revenue for the extended paths (i.e., that $r(\mathcal{P}_1, \mathcal{Q}) - r(\mathcal{P}_2, \mathcal{Q}) = r(\mathcal{P}_1) - r(\mathcal{P}_2)$) meaning that the extension will result in the same increase in revenue for both $\mathcal{P}_1$ and $\mathcal{P}_2$. Again, due to the additivity of the mean and the variance we have that

$$
\mu(\mathcal{P}_1, \mathcal{Q}) = \mu(\mathcal{P}_1) + \mu(\mathcal{Q}) \leq \mu(\mathcal{P}_2) + \mu(\mathcal{Q}) = \mu(\mathcal{P}_2, \mathcal{Q})
$$

and

$$
\sqrt{\sigma^2(\mathcal{P}_1, \mathcal{Q})} = \sqrt{\sigma^2(\mathcal{P}_1) + \sigma^2(\mathcal{Q})} \leq \sqrt{\sigma^2(\mathcal{P}_2) + \sigma^2(\mathcal{Q})} = \sqrt{\sigma^2(\mathcal{P}_2, \mathcal{Q})}
$$

As both $h_T(\cdot, \cdot)$ and $f(\cdot)$ are non-decreasing we have that

$$
f\left(h_T\left(\mu(\mathcal{P}_1, \mathcal{Q}), \sqrt{\sigma^2(\mathcal{P}_1, \mathcal{Q})}\right)\right) \leq f\left(h_T\left(\mu(\mathcal{P}_2, \mathcal{Q}), \sqrt{\sigma^2(\mathcal{P}_2, \mathcal{Q})}\right)\right)
$$

and therefore $C(\mathcal{P}_1, \mathcal{Q}) \leq C(\mathcal{P}_2, \mathcal{Q})$, which concludes the proof.

$\square$