INSEEC Research
PARIS • BORDEAUX • LYON • ALPES-SAVOIE
MONACO • LONDON • CHICAGO

Working Paper
n°: 2015-100-01

Approximate solution of a profit maximization constrained virtual business planning problem

Alexandre Dolgui [a], Sergey Kovalev [b], Erwin Pesch [c]

a Ecole Nationale Supérieure des Mines de Saint-Etienne, LIMOS UMR 6158 CNRS, 158, cours Fauriel, 42023 Saint-Etienne Cedex 2, France
b INSEEC Business School ECE Lyon, 19 place Tolozan, 69001 Lyon, France
c Institute of Information Systems at the University of Siegen, Germany

May 2015

An ulterior version of this article appeared in Omega, The International Journal of Management Science, Volume 57, Issue Part B, December 2015, ISSN 0305-0483
It can be purchased at: http://www.sciencedirect.com/science/article/pii/S0305048315000912

AbstractA virtual business problem is studied, in which a company-contractor outsources production to specialized subcontractors. Finances of the contractor and resource capacities of subcontractors are limited. The objective is to select subcontractors and distribute a part of the demanded production among them so that the profit of the contractor is maximized. A generalization of the knapsack problem, called Knapsack-of-Knapsacks (K-of-K), is used to model this situation, in which items have to be packed into small knapsacks and small knapsacks have to be packed into a large knapsack. A fully polynomial time approximation scheme is developed to solve the problem K-of-K.

Keywords: virtual business planning; knapsack problem; dynamic programming; fully polynomial time approximation scheme.

1 Introduction

Taking advantage of the globalization of world economy, many large companies outsource production to specialized subcontractors. There are several reasons why companies recur to subcontracting: reduction of own labor costs, professional expertise and possession of skills and technologies, which are not retained in-house, flexibility of subcontracting, testing the market as a prelude to the possible production, coverage of possible shortcomings in supply of goods, see Imrie [10] and Balachandran et al. [2]. Subcontracting can be used in a short term (Danese [5]) or in a long-term multi-period production planning (Gomes da Silva et al. [8]).

If the resource capacities of the contractor and subcontractors are limited, the problem of selecting the subcontractors and distributing a part of the demanded production among them with the aim of maximizing the profit of the contractor becomes complicated. We suggest to model this virtual business planning problem as a mathematical programming problem which we call *Knapsack of Knapsacks (K-of-K)*.

Let us formulate the problem K-of-K, and then describe the virtual business planning problem of our prime interest in terms of the problem K-of-K. There are items of F classes to be packed into at most F small knapsacks, which are to be further packed into a large knapsack. Items of the same class i are packed into the same small knapsack i, $i = 1, \ldots, F$. Each item can be packed or not. Class i consists of n_i item types. Items of the same type are identical. The number of copies u_{ij}, a per item profit p_{ij} and a per item capacity consumption c_{ij} are associated with each item type j of class i for all item types and classes. A small knapsack i has a capacity C_i, $i = 1, \ldots, F$, and the large knapsack has a capacity C. It can be assumed without loss of generality that $C_i \leq C$, $i = 1, \ldots, F$. If at least one item of class i is packed, then the small knapsack i is in use, which implies a fixed profit f_i and a fixed extra capacity s_i consumed by this small knapsack in the large knapsack. The problem is to determine the number of items of each type and class to be packed such that the overall profit is maximized, provided that the capacities of the small knapsacks and the large knapsack are not exceeded. All numerical data are non-negative rational numbers.

Problem K-of-K can be formulated as an integer linear programming problem. Denote by x_{ij} the unknown number of items of class i and type j to be packed, and by x the structure with entries x_{ij}, $j = 1, \ldots, n_i$, $i = 1, \ldots, F$. Denote row i of x as $x_i = (x_{i1}, x_{i2}, \ldots, x_{in_i})$. Introduce variables $y_i \in \{0, 1\}$ such that $y_i = 1$ if and only if at least one item of class i

is packed and, hence, small knapsack i is in use, $i = 1, \ldots, F$. Denote $y = (y_1, \ldots, y_F)$, $P_i(x_i) = \sum_{j=1}^{n_i} p_{ij}x_{ij}$ and $C_i(x_i) = \sum_{j=1}^{n_i} c_{ij}x_{ij}$, $i = 1, \ldots, F$.

Problem K-of-K:

$$\max P(x, y) := \sum_{i=1}^{F} \Big(P_i(x_i) + f_i y_i\Big) = \sum_{i=1}^{F} \Big(\sum_{j=1}^{n_i} p_{ij}x_{ij} + f_i y_i\Big), \text{ subject to}$$

$$C_i(x_i) = \sum_{j=1}^{n_i} c_{ij}x_{ij} \le C_i, \quad i = 1, \ldots, F, \tag{1}$$

$$C(x, y) := \sum_{i=1}^{F} \Big(C_i(x_i) + s_i y_i\Big) = \sum_{i=1}^{F} \Big(\sum_{j=1}^{n_i} c_{ij}x_{ij} + s_i y_i\Big) \le C, \tag{2}$$

$$\sum_{j=1}^{n_i} x_{ij} \le y_i \sum_{j=1}^{n_i} u_{ij}, \quad i = 1, \ldots, F, \tag{3}$$

$$y_i \le \sum_{j=1}^{n_i} x_{ij}, \quad i = 1, \ldots, F, \tag{4}$$

$$x_{ij} \in \{0, 1, \ldots, u_{ij}\}, \ y_i \in \{0, 1\}, \ j = 1, \ldots, n_i, \quad i = 1, \ldots, F. \tag{5}$$

Constraints (1) take care of not exceeding the capacities of the small knapsacks, and constraint (2) does it for the large knapsack. Constraints (3) ensure that if the small knapsack i is not in use ($y_i = 0$), then no item can be packed in it, that is, $x_{ij} = 0$ for $j = 1, \ldots, n_i$. Values $\sum_{j=1}^{n_i} u_{ij}$ in these constraints play the role of sufficiently large numbers such that if $y_i = 1$, then values x_{ij} are not bounded from above by these constraints. Constraints (4) guarantee that if $x_{ij} = 0$ for $j = 1, \ldots, n_i$, then the small knapsack i is not used, that is, $y_i = 0$. (5) are box and integrality constraints for the variables.

Problem K-of-K appears as a sub-problem in virtual business planning. Consider a company that fulfills demands for various products. The demand for each product is a given number of identical items to be manufactured. The company uses subcontractors to satisfy the demands. There is a given assignment of products to subcontractors such that a given product can be manufactured by a single given subcontractor. Products assigned to the same subcontractor can be viewed as those of the same class. A product of the same class can be associated with a type, and types of product i can be numbered $j = 1, \ldots, n_i$.

From the side of subcontractor i, manufacturing one item of class i and type j requires an effort that can be expressed in the required workforce capacity which, in turn, can be expressed in salary c'_{ij} to be paid. If subcontractor i produces x_{ij} items of class i and type j, $j = 1, \ldots, n_i$, then the total cost of this production imposed on the contractor is $\sum_{j=1}^{n_i}(1 + h_i)c'_{ij}x_{ij} :=$

$\sum_{j=1}^{n_i} c_{ij}x_{ij}$, where $h_i \cdot 100\%$ are the overheads and the markup of subcontractor i, and c_{ij} is the total cost of producing one item of class i and type j for the contractor. The workforce capacity of subcontractor i is limited, which can be expressed by imposing an upper bound C'_i on the total salary: $\sum_{j=1}^{n_i} c'_{ij}x_{ij} \leq C'_i$, or equivalently, by imposing an upper bound on the total cost for the contractor: $\sum_{j=1}^{n_i} c_{ij}x_{ij} \leq (1+h_i)C'_i := C_i$.

From the side of the contractor, the selling price of one item of class i and type j is p^0_{ij}, hence, the corresponding profit is $p_{ij} = p^0_{ij} - c_{ij}$. Supervising project i by the contractor costs s_i and implies a profit f_i from this supervising work. The total cost that the contractor can afford is limited by C. The contractor would like to select a part of the overall demand and assign it to the selected subcontractors so that the total profit is maximized. The unassigned demand is lost or it can be satisfied in the future by solving the same problem with updated input data.

Let (x^*, y^*) denote an optimal solution of the K-of-K problem and let $P^* = P(x^*, y^*)$. Given $0 < \varepsilon < 1$, algorithm H for problem K-of-K is called a $(1-\varepsilon)$-*approximation algorithm* if it delivers a $(1-\varepsilon)$-*approximate solution* $(x^{(H)}, y^{(H)})$ such that $P(x^{(H)}, y^{(H)}) \geq (1-\varepsilon)P^*$ for all problem instances. A family of approximation algorithms $\{H_\varepsilon\}$ for problem K-of-K is a *Fully Polynomial Time Approximation Scheme (FPTAS)* if, for any given $0 < \varepsilon < 1$, algorithm H_ε is a $(1-\varepsilon)$-approximation algorithm which runs in polynomial time in the problem instance length in binary encoding and $1/\varepsilon$.

Problem K-of-K is a generalization of the NP-hard multiple-choice knapsack problem, see Martello and Toth [18]. Theoretically, an FPTAS is the best algorithm that can be developed for an NP-hard problem (Garey and Johnson [6]). First FPTASes for the classic knapsack problem were developed by Ibarra and Kim [9], Sahni [20], Lawler [15], Gens and Levner [7] and Magazine and Oguz [16], and the most recent advances can be found in the monograph of Kellerer et al. [11]. Computer experiments demonstrated that FPTAS algorithms are competitive to other algorithms, see for example, Martello and Toth [17] and Kovalyov et al. [14]. The number of publications suggesting FPTASes for optimization problems continues to grow, see the latest papers of Kellerer and Strusevich [12], Shabtay et al. [21] and Gafarov et al. [3]. Multicriteria problems are tackled by similar approaches, see Cheng et al. [4], Angel et al. [1], Shabtay et al. [22] and Ramos et al. [19].

This paper presents an FPTAS for the problem K-of-K. The general idea is given in the next section. A subroutine, which is to construct a so-called Δ-lattice of feasible solutions,

is described in Section 3. A dynamic programming algorithm to solve a "rounded problem" K-of-K is presented in Section 4. A procedure to improve lower and upper bounds of the optimal solution value and the general steps of the FPTAS are described in Section 5. The paper completes with a managerial insight, a short summary of the results and suggestions for future research.

2 General idea of FPTAS

Let lower and upper bounds L and U be given such that $0 < L \leq P^* \leq U$. We can set $L = \min_{\forall i,j}\{p_{ij}\}$ and $U = \sum_{\forall i,j} p_{ij} + \sum_{\forall i} f_i$. Calculate a scaling parameter $\Delta = \varepsilon L/(2F)$. The general idea of our FPTAS can be described as follows:

- decomposing problem K-of-K into F subproblems, denoted as (K-of-K)$_i$, such that, for the same optimal solution (x^*, y^*) of the problem K-of-K, row x_i^* is a feasible solution of the subproblem (K-of-K)$_i$, $i = 1, \ldots, F$;

- constructing a Δ-*lattice* for each problem (K-of-K)$_i$. A set $D_i = \{x_i^{(1)}, \ldots, x_i^{(K_i)}\}$ of feasible solutions of problem (K-of-K)$_i$ is called Δ-lattice for this problem if for any feasible solution x_i^0 of this problem such that $P_i(x_i^0) \leq U$, there is a solution x_i' from the Δ-lattice D_i such that $x_i' = (0, \ldots, 0)$ if $x_i^0 = (0, \ldots, 0)$, $x_i' \neq (0, \ldots, 0)$ if $x_i^0 \neq (0, \ldots, 0)$, and, furthermore, $|P_i(x_i') - P_i(x_i^0)| \leq \Delta$ and $C_i(x_i') \leq C_i(x_i^0)$;

- solving a *master problem*, denoted as (K-of-K)-Rou, which is to select $x_i^{(h_i)} \in D_i$, $i = 1, \ldots, F$, such that $(x^{(\varepsilon)}, y^{(\varepsilon)})$ is a $(1-\varepsilon)$-approximate solution for problem K-of-K, where $x^{(\varepsilon)} = (x_1^{(h_1)}, \ldots, x_F^{(h_F)})$, $y_i^{(\varepsilon)} = 1$ if $x_i^{(h_i)} \neq (0, \ldots, 0)$ and $y_i^{(\varepsilon)} = 0$ if $x_i^{(h_i)} = (0, \ldots, 0)$, $i = 1, \ldots, F$.

Let us formulate problems (K-of-K)$_i$ and (K-of-K)-Rou.

Problem (K-of-K)$_i$:

$$\max \quad P_i(x_i) = \sum_{j=1}^{n_i} p_{ij} x_{ij}, \text{ subject to}$$

$$C_i(x_i) = \sum_{j=1}^{n_i} c_{ij} x_{ij} \leq C_i,$$

$$x_{ij} \in \{0, 1, \ldots, u_{ij}\}, \quad j = 1, \ldots, n_i.$$

In Section 3 we will show how to construct Δ-lattice D_i. At this juncture, assume that Δ-lattices D_i, $i = 1, \ldots, F$, have been constructed. Introduce 0-1 variables z_{ij} such that $z_{ij} = 1$ if and only if solution $x_i^{(j)} \in D_i$ is selected as part $x_i^{(h_i)}$ of the solution $x^{(h)} = (x_1^{(h_1)}, \ldots, x_F^{(h_F)})$ for the K-of-K problem. Denote $P_i^{(j)} = C_i(x_i^{(j)}) = 0$ if $x_i^{(j)} = (0, \ldots, 0)$ and $P_i^{(j)} = P_i(x_i^{(j)}) + f_i$, $C_i^{(j)} = C_i(x_i^{(j)}) + s_i$ if $x_i^{(j)} \neq (0, \ldots, 0)$, $i = 1, \ldots, F$, $j = 1, \ldots, K_i$.

Master problem (K-of-K)-Rou:

$$\max \quad R(z) = \sum_{i=1}^{F} \sum_{j=1}^{K_i} \lfloor P_i^{(j)} z_{ij} / \Delta \rfloor, \text{ subject to}$$

$$T(z) = \sum_{i=1}^{F} \sum_{j=1}^{K_i} C_i^{(j)} z_{ij} \leq C,$$

$$\sum_{j=1}^{K_i} z_{ij} = 1, \quad i = 1, \ldots, F,$$

$$z_{ij} \in \{0, 1\}, \quad i = 1, \ldots, F, \ j = 1, \ldots, K_i.$$

Let $z^{(\varepsilon)}$ denote an optimal solution of the master problem (K-of-K)-Rou. Determine $(x^{(\varepsilon)}, y^{(\varepsilon)})$ such that 1) row $x_i^{(\varepsilon)} = x_i^{(j)}$ where j is determined from $z_{ij}^{(\varepsilon)} = 1$, and 2) knapsack i usage variable $y_i^{(\varepsilon)} = 1$ if $x_i^{(j)} \neq (0, \ldots, 0)$, and $y_i^{(\varepsilon)} = 0$, otherwise.

Theorem 1 *Pair $(x^{(\varepsilon)}, y^{(\varepsilon)})$ is a $(1 - \varepsilon)$-approximate solution of the problem K-of-K.*

Proof: Consider a feasible solution (x', y') of the problem K-of-K whose rows x_i' are taken from the Δ-lattice D_i and satisfy $x_i' = (0, \ldots, 0)$ if $x_i^* = (0, \ldots, 0)$, $x_i' \neq (0, \ldots, 0)$ if $x_i^* \neq (0, \ldots, 0)$, and $|P_i(x_i') - P_i(x_i^*)| \leq \Delta$ and $C_i(x_i') \leq C_i(x_i^*)$, $i = 1, \ldots, F$. Further, $y_i' = 0$ if x_i' is a zero vector and $y_i' = 1$ otherwise, $i = 1, \ldots, F$. Such a solution (x', y') exists by the definition of Δ-lattices D_i.

We obtain $y_i' = y_i^*$, $i = 1, \ldots, F$, and, hence,

$$C(x', y') = \sum_{i=1}^{F} \Big(C_i(x_i') + s_i y_i' \Big) \leq \sum_{i=1}^{F} \Big(C_i(x_i^*) + s_i y_i^* \Big) \leq C.$$

Let $x_i' = x_i^{(h_r)}$, i.e., x_i' is numbered h_r-th in the Δ-lattice D_i. Define $z_{ih_r}' = 1$ and $z_{ij}' = 0$ for $j \neq h_r$, $j = 1, \ldots, K_i$. We have

$$\sum_{i=1}^{F} \sum_{j=1}^{K_i} C_i^{(j)} z_{ij}' = \sum_{i=1}^{F} \Big(C_i(x_i') + s_i y_i' \Big) \leq C,$$

which proves that if problem K-of-K has a feasible solution, then problem K-of-K-Rou also has a feasible solution.

From $|P_i(x'_i) - P_i(x^*_i)| \le \Delta$ in the definition of the Δ-lattice D_i, $i = 1, \ldots, F$, we obtain

$$\sum_{i=1}^{F} P_i(x'_i) \ge \sum_{i=1}^{F} P_i(x^*_i) - F\Delta.$$

Since $y'_i = y^*_i$, $i = 1, \ldots, F$, we further obtain

$$\sum_{i=1}^{F} (P_i(x'_i) + f_i y'_i) \ge \sum_{i=1}^{F} (P_i(x^*_i) + f_i y^*_i) - F\Delta. \tag{6}$$

The following chain of relations completes the proof.

$$P(x^{(\varepsilon)}, y^{(\varepsilon)}) = \sum_{i=1}^{F} \sum_{j=1}^{K_i} P_i^{(j)} z_{ij}^{(\varepsilon)} \ge \Delta \sum_{i=1}^{F} \sum_{j=1}^{K_i} \lfloor P_i^{(j)} z_{ij}^{(\varepsilon)} / \Delta \rfloor \ge \Delta \sum_{i=1}^{F} \sum_{j=1}^{K_i} \lfloor P_i^{(j)} z'_{ij} / \Delta \rfloor =$$

$$\Delta \sum_{i=1}^{F} \lfloor (P_i(x'_i) + f_i y'_i)/\Delta \rfloor \ge \sum_{i=1}^{F} (P_i(x'_i) + f_i y'_i) - F\Delta \ge \sum_{i=1}^{F} (P_i(x^*_i) + f_i y^*_i) - 2F\Delta \ge (1-\varepsilon) P^*.$$

Here the next to last relation follows from (6). ∎

3 Constructing Δ-lattice

Based on the problem (K-of-K)$_i$, construct the following *rounded* problem (K-of-K)$_i$-Rou. Calculate scaling parameters $\delta_i = \Delta / n_i$ and values $U_{ij} = \min\{\lceil U/\delta_i \rceil + j, \sum_{r=1}^{j} \lceil p_{ir} u_{ir} / \delta_i \rceil\}$, $j = 1, \ldots, n_i$. Value U_{ij} is an upper bound on the value $\sum_{r=1}^{j} \lceil p_{ir} x_{ir} / \delta_i \rceil$ for any feasible solution x_i of the problem (K-of-K)$_i$ such that $P_i(x_i) \le U$.

Problem (K-of-K)$_i$-Rou:

$$\max \quad P_i^{Rou}(x_i) = \sum_{j=1}^{n_i} \lceil p_{ij} x_{ij} / \delta_i \rceil, \text{ subject to}$$

$$C_i(x_i) = \sum_{j=1}^{n_i} c_{ij} x_{ij} \le C_i,$$

$$x_{ij} \in \Big\{ v_j(0), v_j(1), \ldots, v_j(U_{ij}) \Big\}, \quad j = 1, \ldots, n_i, \text{ where}$$

$$v_j(k) = \min\{x_{ij} \mid x_{ij} \in \{0, 1, \ldots, u_{ij}\}, \ \lceil p_{ij} x_{ij} / \delta_i \rceil = k\}.$$

Since $\lceil p_{ij} x_{ij} / \delta_i \rceil = k$ is equivalent to $(k-1)\delta_i < p_{ij} x_{ij} \le k\delta_i$ and further to $(k-1)\delta_i / p_{ij} < x_{ij} \le k\delta_i / p_{ij}$, we obtain

$$v_j(0) = 0 \text{ and}$$

$$v_j(k) = \begin{cases} \max\{1, \lceil (k-1)\delta_i / p_{ij} \rceil\}, & \text{if } k\delta_i / p_{ij} \ge 1, \\ \emptyset, & \text{if } k\delta_i / p_{ij} < 1, \end{cases} \quad k = 1, \ldots, U_{ij}.$$

We will describe a dynamic programming algorithm for problem (K-of-K)$_i$-Rou that constructs a Δ-lattice D_i. In this algorithm, denoted as DP$_i$, we recursively compute the value of

$G_j(f)$ which is the minimum value of $C_i(x_i)$, provided that the values for the first j variables $x_{i1}, \ldots, x_{ij}$ are determined and $\sum_{r=1}^{j} \lceil p_{ir}x_{ir}/\delta_i \rceil = f$.

Algorithm DP$_i$ (constructs Δ-lattice D_i)

Step 1 (Initialization) Set $G_j(f) = 0$ for $(f, j) = (0, 0)$, and $G_j(f) = \infty$ for $(f, j) \neq (0, 0)$.

Step 2 (Recursion) For $j = 1, \ldots, n_i$ and $f = 0, 1, \ldots, U_{ij}$, compute the following:

$$G_j(f) = \min_{x_{ij} \in \{v_j(0), \ldots, v_j(f)\}} \{G_{j-1}(f - \lceil p_{ij}x_{ij}/\delta_i \rceil) + c_{ij}x_{ij}\} \tag{7}$$

Step 3 (Δ-lattice D_i) For $f = 0, 1, \ldots, U_{i,n_i}$, if $G_{n_i}(f) \leq C_i$, then find the corresponding feasible solution x_i' with value $P_i^{Rou}(x_i') = f$ by backtracking and include it into D_i.

Since $U_{ij} \leq \lceil U/\delta_i \rceil + j$ for all j, the time complexity of the algorithm DP$_i$ is $O(n_i(U/\delta_i)^2) = O((U/L)^2 n_i^3 F^2/\varepsilon^2)$.

Theorem 2 *Algorithm DP$_i$ constructs a Δ-lattice D_i for the problem (K-of-K)$_i$.*

Proof: Consider an arbitrary feasible solution x_i^0 of the problem(K-of-K)$_i$ such that $P_i(x_i^0) \leq U$. By the definition, it is feasible for the problem (K-of-K)$_i$-Rou. Let $P_i^{Rou}(x_i^0) = f$. Then, by the definition of algorithm DP$_i$, in its Step 3 the backtracking procedure will find x_i' such that $P_i^{Rou}(x_i') = f$ and $C_i(x_i') \leq C_i(x_i^0) \leq C_i$.

If $x_i^0 = (0, \ldots, 0)$, then $f = 0$ and the only solution of the recursive equations (7) is $x_i' = x_i^0 = (0, \ldots, 0)$. If $x_i^0 \neq (0, \ldots, 0)$, then $f \geq 1$, and all solutions x_i' of the recursive equations (7) satisfy $x_i' \neq (0, \ldots, 0)$.

Since the chain of relations

$$f = P_i^{Rou}(x_i') = \sum_{j=1}^{n_i} \left\lceil \frac{p_{ij}x_{ij}'}{\delta_i} \right\rceil < \sum_{j=1}^{n_i} \frac{p_{ij}x_{ij}'}{\delta_i} + n_i \leq \sum_{j=1}^{n_i} \left\lceil \frac{p_{ij}x_{ij}'}{\delta_i} \right\rceil + n_i = f + n_i$$

implies $f\delta_i - n_i\delta_i < \sum_{j=1}^{n_i} p_{ij}x_{ij}' \leq f\delta_i$, and the chain of relations

$$f = P_i^{Rou}(x_i^0) = \sum_{j=1}^{n_i} \left\lceil \frac{p_{ij}x_{ij}^0}{\delta_i} \right\rceil < \sum_{j=1}^{n_i} \frac{p_{ij}x_{ij}^0}{\delta_i} + n_i \leq \sum_{j=1}^{n_i} \left\lceil \frac{p_{ij}x_{ij}^0}{\delta_i} \right\rceil + n_i = f + n_i$$

implies $f\delta_i - n_i\delta_i < \sum_{j=1}^{n_i} p_{ij}x_{ij}^0 \leq f\delta_i$, we obtain $|P_i(x_i') - P_i(x_i^0)| \leq n_i\delta_i = \Delta$, as it is required.

■

Since one solution is found for each objective function value $f = 0, 1, \ldots, U_{in_i}$, the cardinality K_i of the Δ-lattice D_i can be estimated as $K_i \leq \lceil U/\delta_i \rceil + n_i + 1 \leq O(FUn_i/(L\varepsilon))$.

4 Solving problem (K-of-K)-Rou

Problem (K-of-K)-Rou is a rounded variant of the multiple-choice knapsack problem, see Martello and Toth [18], with binary variables z_{ij}. It can be solved by the following dynamic programming algorithm, which we denote as DP. In this algorithm, we recursively compute the value of $H_i(f)$ which is the minimum value of $T(z)$, provided that the values z_{rj} are determined for $r = 1, \ldots, i$ and $j = 1, \ldots, K_i$, and $R(z) = f$.

Algorithm DP (solves problem (K-of-K)-Rou)

Step 1 (Initialization) Set $H_i(f) = 0$ for $(f, i) = (0, 0)$ and $H_i(f) = \infty$ for $(f, i) \neq (0, 0)$.

Step 2 (Recursion) For $f = 0, 1, \ldots, \lfloor U/\Delta \rfloor$ and $i = 1, \ldots, F$, compute the following:

$$H_i(f) = \min_{j=1,\ldots,K_i} \{H_{i-1}(f - \lfloor P_i^{(j)}/\Delta \rfloor) + C_i^{(j)}\}.$$

If the above minimum is attained at an index j, then in the corresponding solution $z_{ij} = 1$ and $z_{ik} = 0$ for $k \neq j$.

Step 3 (Optimal solution) Determine optimal solution value $P^{(\varepsilon)} = \max\{f | H_F(f) \leq C\}$ and find the corresponding optimal solution $z^{(\varepsilon)}$ by backtracking.

The time complexity of the algorithm DP can be evaluated as $O(U \sum_{i=1}^{F} K_i/\Delta)$. Since $\Delta = \varepsilon L/(2F)$ and $K_i \leq O(FUn_i/(L\varepsilon))$, it is $O((U/L)^2 F^2 \sum_{i=1}^{F} n_i/\varepsilon^2)$. Thus, for given L, U and ε, the run time of our $(1-\varepsilon)$-approximation algorithm for the problem K-of-K is determined by the total run time of the algorithms DP_i, $i = 1, \ldots, F$, which is $O((U/L)^2 (\sum_{i=1}^{F} n_i)^3 F^2/\varepsilon^2)$.

5 Improving lower and upper bounds and general steps of the FPTAS

The run time of our $(1-\varepsilon)$-approximation algorithm for the problem K-of-K can be reduced to $O((\sum_{i=1}^{F} n_i)^3 F^2/\varepsilon^2 + (\sum_{i=1}^{F} n_i)^3 F^2 \log(U/L))$ by employing bound improvement procedure similar to that presented by Kovalyov [13] for a minimization problem. The procedure, which we call BIP-Max(L, U) (Bound Improvement Procedure for Maximization problem with bounds L and U), finds P_0 such that one can re-set $L := P_0$ and $U := 4P_0$.

Procedure BIP-Max(L, U) (given L and U such that $0 < L \leq P^* \leq U$, finds P_0 such that $0 < P_0 \leq P^* \leq 4P_0$)

Step 1 Find integer k such that $2^k L \leq U \leq 2^{k+1} L$. Set $L_0 = L$ and $i = 1$.

Step 2 Apply $(1-\varepsilon)$-approximation algorithm for the problem K-of-K with parameters $L := 2^i L_0$, $U := 2^{i+1} L_0$ and $\varepsilon := 1/2$. If $(1-\varepsilon)$-approximate solution is found, denote its value as $P^{(i)}$. If it is not found, set $P^{(i)} = -\infty$. If $i = k$, then go to Step 3. If $i < k$, then re-set $i := i + 1$ and repeat Step 2.

Step 3 Calculate $P_0 = \max_{0 \leq i \leq k} \{P^{(i)}\}$.

Theorem 3 *Procedure BIP-Max(L, U) finds P_0 such that $0 < P_0 \leq P^* \leq 4P_0$. Its run time is $O((\sum_{i=1}^{F} n_i)^3 F^2 \log(U/L))$.*

Proof: Observe that $2^{i^*} L_0 \leq P^* \leq 2^{i^*+1} L_0$ for $0 \leq i^* \leq k$ and the $(1-\varepsilon)$-approximation algorithm for the problem K-of-K with parameters $L := L' = 2^{i^*} L_0$, $U := 2^{i^*+1} L_0 = 2L'$ and $\varepsilon := 1/2$ will find a solution with value $P^{(i^*)}$ such that $P^{(i^*)} \geq P^* - \varepsilon L' = P^* - L'/2 \geq L'/2 > 0$. Then $0 < L'/2 \leq P^{(i^*)} \leq P^* \leq 2L'$ and $0 < P^{(i^*)} \leq P_0 \leq P^* \leq 2L' \leq 4P^{(i^*)} \leq 4P_0$, as required for the first statement of the theorem.

Since $k \leq \log(U/L)$ and the run time of the $(1-\varepsilon)$-approximation algorithm for the problem K-of-K with parameters $U = 2L$ and $\varepsilon := 1/2$ is $O((\sum_{i=1}^{F} n_i)^3 F^2)$, the run time of BIP-Max(L, U) is $O((\sum_{i=1}^{F} n_i)^3 F^2 \log(U/L))$. ∎

Our FPTAS for the problem K-of-K can be described as follows.

Step 1 Determine L and U such that $0 < L \leq P^* \leq U$.

Step 2 Apply BIP-Max(L, U) to find P_0 such that $0 < P_0 \leq P^* \leq 4P_0$.

Step 3 Apply $(1-\varepsilon)$-approximation algorithm with $L = P_0$ and $U = 4P_0$.

Main steps of the $(1-\varepsilon)$-approximation algorithm:

Step 3.1 Apply algorithms DP$_i$, $i = 1, \ldots, F$, and construct Δ-lattices $D_1, \ldots, D_F$.

Step 3.2 Formulate and solve problem (K-of-K)-Rou.

Step 3.3 Based on the solution $z^{(\varepsilon)}$ of the problem (K-of-K)-Rou, determine $(1-\varepsilon)$-approximate solution $(x^{(\varepsilon)}, y^{(\varepsilon)})$ of the problem K-of-K.

Our FPTAS can be used to solve the problem K-of-K optimally. If all numerical parameters of this problem are non-negative integer numbers, then we can apply algorithms DP_i, $i = 1, \ldots, F$, and DP in which $\Delta = \delta_1 = \cdots = \delta_F = 1$. This application will produce an optimal solution for the problem K-of-K, if it exists, in $O(U^2 \sum_{i=1}^{F} n_i)$ time, which is pseudo-polynomial.

6 Managerial insight and conclusions

If a company would like to take advantage of the outsourcing for manufacturing a number of demanded products, it needs to account for the limited resource capacities of the subcontractors and its own limited finances. A mathematical interpretation of the corresponding profit maximization virtual business planning problem results in a generalization of the classic knapsack problem to the problem K-of-K. This 0-1 linear programming problem can be solved by employing a commercial software, which is based on the exponential enumeration techniques. Alternatively, an FPTAS developed in this paper can be used. It guarantees a given relative error ε of the solution not to be exceeded and runs in time polynomial in the problem instance length and $1/\varepsilon$. Specifically, it runs in $O((\sum_{i=1}^{F} n_i)^3 F^2/\varepsilon^2 + (\sum_{i=1}^{F} n_i)^3 F^2 \log(U/L))$ time. The FPTAS can be used to solve the problem K-of-K optimally in $O(U^2 \sum_{i=1}^{F} n_i)$ time.

Further research can concentrate on the modifications of this problem to include other criteria and constraints that address practical conditions of virtual business planning problems. Improving efficiency of the presented FPTAS is worth of investigation as well.

References

[1] Angel E, Bampis E, Kononov A. On the approximate tradeoff for bicriteria batching and parallel machine scheduling problems. Theoretical Computer Science 2003; 306(1-3): 319–338.

[2] Balachandran KR, Wang HW, Li SH, Wang T. In-house capability and supply chain decisions. Omega 2013; 41(2): 473–484.

[3] Gafarov ER, Dolgui A, Werner F. A new graphical approach for solving single-machine scheduling problems approximately. International Journal of Production Research 2014; 52(13): 3762–3777.

[4] Cheng TCE, Janiak A, Kovalyov MY. Bicriterion single machine scheduling with resource dependent processing times. SIAM Journal on Optimization 1998; 8(2): 617–630.

[5] Danese P. Supplier integration and company performance: A configurational view. Omega 2013; 41(6): 1029–1041.

[6] Garey MR, Johnson DS. Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman and Co., San Francisco: 1979.

[7] Gens GV, Levner EV. Fast approximation algorithms for knapsack type problems. In K. Iracki, K. Malinowski, and S. Walukiewicz, editors, Optimization Techniques, Part 2, volume 74 of Lecture Notes in Control and Information Sciences, pages 185–194. Springer, 1980.

[8] Gomes da Silva C, Figuera J, Lisboa J, Barman S. An interactive decision support system for an aggregate production planning model based on multiple criteria mixed integer linear programming. Omega 2006; 34(2): 167–177.

[9] Ibarra OH, Kim CE. Fast approximation algorithm for the knapsack and sum of subset problems. Journal of the ACM 1975; 22(4): 463–468.

[10] Imrie R. 'A strategy of the last resort'? Reflections on the role of the subcontract in the United Kingdom. Omega 1994; 22(6): 569–578.

[11] Kellerer H, Pferschy U, Pisinger D. Knapsack problems. Springer, Berlin: 2004.

[12] Kellerer H, Strusevich V. Fast approximation schemes for Boolean programming and scheduling problems related to positive convex Half-Product. European Journal of Operational Research 2013; 228(1): 24–32.

[13] Kovalyov MY. Improving the complexities of approximation algorithms for optimization problems. Operations Research Letters 1995; 17(2): 85–87.

[14] Kovalyov MY, Portmann M-C, Oulamara A. Optimal testing and repairing a failed series system. Journal of Combinatorial Optimization 2006; 12: 279–295.

[15] Lawler EL. Fast approximation algorithms for knapsack problems. Mathematics of Operations Research, 1979; 4: 339–356.

[16] Magazine MY, Oguz O. A fully polynomial approximation algorithm for the 0-1 knapsack problem. European Journal of Operational Research, 1981; 8:270–273.

[17] Martello S, Toth P. Approximation schemes for the subset-sum problem: survey and experimental analysis. European Journal of Operations Research 1985; 22(1): 56–69.

[18] Martello S, Toth P. Knapsack problems: algorithms and computer implementations. Wiley: 1990.

[19] Ramos TRP, Gomes MI, Barbosa-Povoa AP. Planning a sustainable reverse logistics system: Balancing costs with environmental and social concerns. Omega 2014; 48: 60–74.

[20] Sahni S. General techniques for combinatorial approximation. Operations Research 1977; 25(6): 920–936.

[21] Shabtay D, Arviv K, Stern H, Edan Y. A combined robot selection and scheduling problem for flow-shops with no-wait restrictions. Omega 2014; 43: 96–107.

[22] Shabtay D, Bensoussan Y, Kaspi M. A bicriteria approach to maximize the weighted number of just-in-time jobs and to minimize the total resource consumption cost in a two-machine flow-shop scheduling system. International Journal of Production Economics 2012; 136(1): 67–74.