Omega 57 (2015) 212–216

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

Approximate solution of a profit maximization constrained virtual business planning problem ☆

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

[a] Ecole 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

ARTICLE INFO

Article history:
Received 1 July 2014
Accepted 2 May 2015
Available online 16 May 2015

Keywords:
Virtual business planning
Knapsack problem
Dynamic programming
Fully polynomial time approximation scheme

ABSTRACT

A 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.

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 [11] and Balachandran et al. [2]. Subcontracting can be used in a short term [5] or in a long-term multi-period production planning [8], there are different subcontracting price schemes [24], and there can be benefits as well as drawbacks of having one or several subcontractors [9].

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, $i=1,\dots,F$. 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, $i=1,\dots,F$, and the large knapsack has a capacity C. It can be assumed without loss of generality that $C_i \le C$, $i=1,\dots,F$. 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}, $j=1,\dots,n_i$, $i=1,\dots,F$. Denote row i of x as $x_i=(x_{i1},x_{i2},\dots,x_{in_i})$. 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, $i=1,\dots,F$. Denote $y=(y_1,\dots,y_F)$, $P_i(x_i)=\sum_{j=1}^{n_i} p_{ij}x_{ij}$ and $C_i(x_i)=\sum_{j=1}^{n_i} c_{ij}x_{ij}$, $i=1,\dots,F$.

Problem K-of-K:

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

☆This manuscript was processed by Associate Editor Kovalyov.

* Corresponding author.
E-mail addresses: dolgui@emse.fr (A. Dolgui), skovalev@inseec.com (S. Kovalev), erwin.pesch@uni-siegen.de (E. Pesch).

http://dx.doi.org/10.1016/j.omega.2015.05.001
0305-0483/

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

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

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

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

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

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 the 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, $y_i = 0$. Eqs. (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: $\sum_{j=1}^{n_i} c'_{ij}x_{ij} \le C'_i$, or equivalently, by imposing an upper bound on the total cost for the contractor: $\sum_{j=1}^{n_i} c_{ij}x_{ij} \le (1 + h_i)C'_i := C_i$. 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 $P(x^{(H)}, y^{(H)}) \ge (1-\varepsilon)P^*$ 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 [19]. Theoretically, an FPTAS is the best algorithm that can be developed for an NP-hard problem [6]. First FPTASes for the classic knapsack problem were developed by Ibarra and Kim [10], Sahni [21], Lawler [16], Gens and Levner [7] and Magazine and Oguz [17], and the most recent advances can be found in the monograph of Kellerer et al. [12]. Computer experiments demonstrated that FPTAS algorithms are competitive to other algorithms, see for example, Martello and Toth [18] and Kovalyov et al. [15]. The number of publications suggesting FPTASes for optimization problems continues to grow, see the latest papers of Kellerer and Strusevich [13], Shabtay et al. [22] and Gafarov et al. [3]. Multi-criteria problems are tackled by similar approaches, see Cheng et al. [4], Angel et al. [1], Shabtay et al. [23] and Ramos et al. [20].

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

- introducing F subproblems, denoted as $(\text{K-of-K})_i$, 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 $(\text{K-of-K})_i$, $i = 1, \ldots, F$;
- constructing a Δ-*lattice* for each problem $(\text{K-of-K})_i$. A set $D_i = \{x_i^{(1)}, \ldots, x_i^{(K_i)}\}$ of feasible solutions of problem $(\text{K-of-K})_i$ is called Δ-lattice for this problem if for any feasible solution x^0_i of this problem such that $P_i(x^0_i) \le U$, there is a solution x'_i from the Δ-lattice D_i such that $x'_i = (0, \ldots, 0)$ if $x^0_i = (0, \ldots, 0)$, $x'_i \ne (0, \ldots, 0)$ if $x^0_i \ne (0, \ldots, 0)$, and, furthermore, $|P_i(x'_i) - P_i(x^0_i)| \le \Delta$ and $C_i(x'_i) \le C_i(x^0_i)$;
- 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 $x_i^{(h_i)} \ne (0, \ldots, 0)$ and $y_i^{(\varepsilon)} = 0$ if $x_i^{(h_i)} = (0, \ldots, 0)$, $i = 1, \ldots, F$.

Let us formulate problems $(\text{K-of-K})_i$ and (K-of-K)-Rou.

Problem $(\text{K-of-K})_i$:

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

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

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

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 $x_i^{(j)} \ne (0, \ldots, 0)$, $i = 1, \ldots, F$, $j = 1, \ldots, K_i$.