DEPARTMENT OF APPLIED ECONOMICS

RESEARCH REPORT

EXACT ALGORITHMS FOR PROCUREMENT PROBLEMS UNDER A TOTAL QUANTITY DISCOUNT STRUCTURE

D.R. GOOSSENS • A.J.T. MAAS • F.C.R. SPIEKSMA • J.J. VAN DE KLUNDERT

OR 0452

KATHOLIEKE UNIVERSITEIT LEUVEN

Exact Algorithms for Procurement Problems under a Total Quantity Discount Structure

D.R. Goossens [(1)], A.J.T. Maas [(2)],
F.C.R. Spieksma [(1)], J.J. van de Klundert [(3)]

(1) Department of Applied Economics, Katholieke Universiteit Leuven, Belgium

(2) RIKS BV, Maastricht, The Netherlands

(3) Department of Mathematics, Maastricht University, The Netherlands

Abstract

In this paper, we study the procurement problem faced by a buyer who needs to purchase a variety of goods from suppliers applying a so-called total quantity discount policy. This policy implies that every supplier announces a number of volume intervals and that the volume interval in which the total amount ordered lies determines the discount. Moreover, the discounted prices apply to all goods bought from the supplier, not only to those goods exceeding the volume threshold. We refer to this cost-minimization problem as the TQD problem. We give a mathematical formulation for this problem and argue that not only it is NP-hard, but also that there exists no polynomial-time approximation algorithm with a constant ratio (unless $P = NP$). Apart from the basic form of the TQD problem, we describe three variants. In a first variant, the market share that one or more suppliers can obtain is constrained. Another variant allows the buyer to procure more goods than strictly needed, in order to reach a lower total cost. In a third variant, the number of winning suppliers is limited. We show that the TQD problem and its variants can be solved by solving a series of min-cost flow problems. Finally, we investigate the performance of three exact algorithms (min-cost flow based branch-and-bound, linear programming based branch-and-bound, and branch-and-cut) on randomly generated instances involving 50 suppliers and 100 goods. It turns out that even the large instances of the basic problem are solved to optimality within a limited amount of time. However, we find that different algorithms perform best in terms of computation time for different variants.

Keywords: procurement, volume discounts, exact algorithm, complexity, min-cost flow, reverse auction

1 Introduction

It is a widespread economic phenomenon that the price of a good depends - among many other things - on the amount ordered. Indeed, there are many reasons for suppliers to offer discounts based on the volume sold to a buyer. Consequently, when it comes to procuring amounts of different goods from different suppliers, it makes sense to consider various alternatives. In fact, choosing

the right suppliers to deliver the right products has become a major concern in many large companies. Reliability, quality, and price are important criteria that guide the choice for suppliers. Moreover, the ever-increasing opportunities that e-commerce and web-based procurement offer for dealing with procurement issues, explain the increased usage of so-called reverse auctions. While traditional auctions involve a single seller and multiple buyers, a reverse auction involves multiple sellers that express bids to provide goods or services and one buyer that chooses the best bids.

In this work, we investigate a basic procurement problem from the viewpoint of a buyer who faces different suppliers that offer a variety of goods using specific discount policies. The discount policy we investigate is one where the supplier has specified a number of volume intervals, and the price per item depends on the volume interval in which the total amount ordered lies. Obviously, a supplier is assumed not to increase its prices in a higher interval. This structure is called total quantity discount (TQD). Furthermore, the prices apply to all units bought from the supplier, which is called an all-unit discount policy (a discussion and classification of various quantity discount policies can be found in Munson and Rosenblatt (1998)). We assume that there are no differences between suppliers other than the prices they charge for the different goods. Thus, the quality of the goods they deliver is assumed to be the same. Given a final demand for each good, the TQD problem is to satisfy demand against minimal cost.

Procurement problems involving discount policies have been studied by many authors. Katz et al. (1994) (see also Sadrian and Yoon (1994)) discuss a procurement problem where they distinguish between purchases on a commitment basis and purchases on an as-ordered basis. They stress the importance of sourcing flexibility and model explicitly the fact that not all future goods should be purchased via committed contracts. In addition, they explicitly consider the number of vendors for each good, and the percentages of the total supply given to each of the vendors. In their discount policy, a supplier discounts the price of each item by the same percentage based on the total dollar value of all goods purchased from the supplier, whereas our policy allows a different discount percentage for each good.

Crama et al. (2004) investigate another procurement problem, characterized by a discount policy very similar to the one used here, in the sense that it also expresses the discount as a function of the total quantity of goods purchased. However, it also differs since it uses one single discount rate for all products. Furthermore, Crama et al. face the additional problem of deciding how to use the purchased raw materials to manufacture the desired quantities of the end-products.

A lot of research on quantity discount policies has been done in the context of lot sizing problems (see e.g. Xu et al. (2000)). Lot sizing problems however deal with when to order what amount of goods and include inventory costs, whereas in this work we want to determine what to order from whom and assume a single-period perspective.

The TQD problem can also be viewed in the context of combinatorial auctions. Combinatorial auctions are relevant when the value of a set of goods is not equal to the sum of the values of the individual goods. Then there are so-called complementary or substitute-effects, and in such a setting it can be beneficial to consider pricing sets of goods instead of pricing only individual goods. The discount policy described above is a way to price a set of goods: the cardinality of the set of all goods ordered determines in which interval the buyer is, and the all-unit discount policy leads to prices that imply complementary effects.

Bichler et al. (2004) outline a classification of allocation problems based on the number of participants and the type of traded goods. According to this classification, the TQD problem is an n-bilateral allocation problem, since there are only two types of participants, i.e., buyers and sellers. In our case, there is only one buyer, which makes it a single-sided auction. Furthermore, the TQD problem is characterized by single-attribute, multi-item, multi-unit bids, because bids can be made on any quantity of a number of heterogeneous goods and all other attributes besides the price are predefined.

Davenport and Kalagnanam (2002) report on a volume discount auction in which discounts are based on quantities for each individual good. Furthermore, they use an incremental discount policy, meaning that the discounts apply only to the additional units above the threshold of the volume interval. Hohner et al. (2003) describe a web-based implementation of this procurement auction at Mars Incorporated.

Eso et al. (2001) also elaborate on the work of Davenport and Kalagnanam. They study a volume discount auction with piece-wise linear supply curves, allowing discontinuities and all-unit discounts. However, they do require additive separable supply curves, which boils down to assuming that the prices charged by a supplier for different commodities are independent. This makes their problem not truly combinatorial, since synergies or substitutability between different goods cannot be reflected in the total price charged by the suppliers. As a result, a total quantity discount structure is not possible in their setting. The authors formulate a column generation based heuristic that provides near-optimal solutions to the bid evaluation problem.

Another procurement auction with marginal-decreasing piecewise-constant supply curves is described in Kothari et al. (2003). This auction also allows all-unit discounts, but it deals only with a single good. Kothari et al. present fully polynomial-time approximation schemes for the winner determination problem and the computation of the Vickrey-Clarke-Groves payments of this auction.

The TQD problem is also related to the so-called deal splitting problem introduced by Shachnai et al. (2004). In this problem, a buyer needs to split an order of multiple units from a set of heterogeneous goods among a set of sellers, each having bounded amounts of the goods, so as to minimize the total cost of the deal. Two variants of the deal splitting problem can be discerned, depending on whether the seller offers packages containing combinations of the goods or whether the buyer can generate such combinations using seller-specified price tables. Shachnai et al. show that for both variants an exact solution can be

found in pseudo-polynomial time if the number of heterogeneous goods is fixed. Moreover, they develop polynomial-time approximation schemes for several subclasses of instances of practical interest.

We now describe shortly the practical application that originally motivated this problem (see Van de Klundert et al. (2003)). Consider a telecommunication company that needs to acquire capacity to accommodate its international calls. This capacity is offered by various so-called carriers, i.e., for each destination, each carrier offers capacity, priced in eurocents per minute. Prices of carriers differ, and - which is particularly relevant for our setting - each carrier uses an interval structure to arrive at a certain price. In other words, the total amount of call-minutes handled by a certain carrier determines the price. The problem is to acquire the right amount of capacity for each destination at minimal cost.

We give the following results. We show that no polynomial-time algorithm for the TQD problem can achieve a constant worst-case ratio (unless $P = NP$). Then, we prove that (a generalization of) the linear programming relaxation of a straightforward formulation of the problem can be solved by min-cost flow. Thus, we prove that a combinatorial algorithm solves the LP-relaxation of the TQD problem. Furthermore, we extend the basic TQD problem by adding some side constraints. Finally, we perform computational experiments comparing three exact algorithms: a min-cost flow based branch-and-bound approach (using the network solver of Ilog Cplex 8.1), a linear programming based branch-and-bound approach (using the MIP solver of Ilog Cplex 8.1) and a branch-and-cut approach (also using the MIP solver of Ilog Cplex 8.1). Section 2 presents the mathematical formulation of our problem, section 3 describes the theoretical results, and section 4 presents three variants of the TQD problem. In section 5, the exact algorithms for the TQD problem and its variants are described and finally section 6 gives our computational results.

2 Mathematical formulation

To state a mathematical formulation of the TQD problem, we use the following notation. We define G as the set of m goods, indexed by k, and S as the set of n suppliers, indexed by i. For each good k in G, we define d_k as the amount of good k to be procured. To each supplier i in S we associate a sequence of intervals $Z_i = \{0, 1, ..., max_i\}$, indexed by j. Furthermore, for each supplier $i \in S$ and interval $j \in Z_i$, l_{ij} and u_{ij} define the minimum and maximum number of goods respectively that needs to be ordered from supplier i to be in interval j. Finally, for each supplier $i \in S$, for each interval $j \in Z_i$ and each good $k \in G$, let c_{ijk} be the price for one item of good k purchased from supplier i in its j-th interval.

We assume that these parameters satisfy the following assumptions:

$$\forall i \in S, j \neq j' \in Z_i : [l_{ij}, u_{ij}) \cap [l_{ij'}, u_{ij'}) = \varnothing, \tag{1}$$
$$\forall i \in S, j \in Z_i \setminus \{max_i\}, k \in G : c_{ijk} \geqslant c_{i,j+1,k}, \tag{2}$$
$$\forall i \in S, j \in Z_i, k \in G : c_{ijk} \geqslant 0,\ l_{ij} \geqslant 0,\ u_{ij} \geqslant 0,\ d_k \geqslant 0. \tag{3}$$

Assumption (1) states that a supplier's intervals should not overlap. The requirement that prices should not increase from one interval to the next is expressed in the second assumption. The last assumption reflects that all prices and all quantities ordered are nonnegative.

We define the decision variable x_{ijk} as the amount of good k purchased from supplier i in interval j. Further, we define a binary decision variable y_{ij} which is 1 if interval j is selected for supplier i and 0 otherwise. This leads to the following formulation of the TQD problem, referred to as TQDF.

minimize

$$\sum_{i \in S} \sum_{j \in Z_i} \sum_{k \in G} c_{ijk} x_{ijk} \tag{4}$$

subject to

$$\sum_{i \in S} \sum_{j \in Z_i} x_{ijk} = d_k \qquad \forall k \in G \tag{5}$$
$$\sum_{j \in Z_i} y_{ij} \leqslant 1 \qquad \forall i \in S \tag{6}$$
$$\sum_{k \in G} x_{ijk} - y_{ij} l_{ij} \geqslant 0 \qquad \forall i \in S, j \in Z_i \tag{7}$$
$$\sum_{k \in G} x_{ijk} - y_{ij} u_{ij} \leqslant 0 \qquad \forall i \in S, j \in Z_i \tag{8}$$
$$x_{ijk} \geqslant 0 \qquad \forall i \in S, j \in Z_i, k \in G \tag{9}$$
$$y_{ij} \in \{0, 1\} \qquad \forall i \in S, j \in Z_i \tag{10}$$

The objective function (4) states that the amount of goods k ordered from supplier i when in interval j, times the corresponding price must be minimal. Constraints (5) make sure that the demand for each good is met, while constraints (6) guarantee that at most one interval per supplier is selected. Constraints (7) and (8) ensure that if an interval j is selected, the total amount of goods purchased from supplier i is between the bounds of that interval. If an interval j is not selected, these constraints ensure that $x_{ijk} = 0$. Constraints (9) state that only a nonnegative amount can be purchased, while constraints (10) define y as a boolean variable. Notice that this formulation allows to order nothing from a supplier. Notice also that we do not require integrality of the x-variables; if the demands and the lower and upper bounds of each volume interval are integral however, then, assuming the existence of a feasible solution, there always exists an optimal solution of TQDF with integral x-values (see section 3).

3 Properties of the TQD problem

In this section we establish the complexity of the TQD problem (section 3.1). We also show that the the LP-relaxation of TQDF can be solved by solving a min-cost flow problem (section 3.2).

3.1 On the complexity of the TQD problem

We show that the TQD problem is a hard problem to solve when aiming for optimal solutions.

Theorem 1 *The decision version of the TQD problem is strongly NP-complete.*

In fact, we can also make the following statement on the approximability of the TQD problem:

Theorem 2 *No polynomial-time approximation algorithm with constant worst-case ratio exists for the TQD problem (unless $P = NP$).*

Next, consider the TQD problem, where - instead of prices for all intervals for each supplier - only prices for the first interval and a discount rate is given. This discount rate δ determines the price $c_{i,j,k}$ of good k in interval j as a function of the price in interval $j-1$ as follows:

$$c_{i,j,k} = (1-\delta)c_{i,j-1,k} \qquad \forall i,k \text{ and } \forall j > 1 \tag{11}$$

We claim that this special case of the TQD problem is still a hard problem.

Theorem 3 *The decision version of the TQD problem with a common discount rate δ is strongly NP-complete.*

Finally, consider the variant of the the TQD problem where the amounts purchased must be *at least as large* as the demands d_k. In such a setting, it might happen that buying more than what is strictly needed reduces the total cost. We refer to this problem as the more-for-less variant of the TQD problem (see section 4.2). We claim that this variant remains a hard problem.

Theorem 4 *The decision version of the more-for-less variant of TQD problem is strongly NP-complete.*

For the proofs of Theorems 1, 2, 3, and 4, we refer to the appendix.

3.2 Min-cost flow and the TQD problem

We now show that the the LP-relaxation of TQDF can be solved by solving a min-cost flow problem. In fact, even in the more general case where for some suppliers intervals are prespecified, the LP-relaxation of the resulting model can still be found by solving a min-cost flow problem.

Let us first state a model which assumes that for an arbitrary given subset of suppliers, referred to as D ($D \subseteq S$), an interval, say $s(i) \in Z_i$, has been selected,

while for the remaining suppliers no interval has been selected. We refer to the following formulation as GENTQDF.

minimize

$$\sum_{i\in S}\sum_{j\in Z_i}\sum_{k\in G} c_{ijk}x_{ijk} \tag{12}$$

subject to

$$\sum_{i\in S}\sum_{j\in Z_i} x_{ijk} = d_k \qquad \forall k \in G \tag{13}$$

$$\sum_{j\in Z_i} y_{ij} \leqslant 1 \qquad \forall i \in S \setminus D \tag{14}$$

$$\sum_{k\in G} x_{ijk} - y_{ij}l_{ij} \geqslant 0 \qquad \forall i \in S \setminus D, j \in Z_i \tag{15}$$

$$\sum_{k\in G} x_{ijk} - y_{ij}u_{ij} \leqslant 0 \qquad \forall i \in S \setminus D, j \in Z_i \tag{16}$$

$$\sum_{k\in G} x_{i,s(i),k} - l_{i,s(i)} \geqslant 0 \qquad \forall i \in D \tag{17}$$

$$\sum_{k\in G} x_{i,s(i),k} - u_{i,s(i)} \leqslant 0 \qquad \forall i \in D \tag{18}$$

$$x_{ijk} \geqslant 0 \qquad \forall i \in S \setminus D, j \in Z_i, k \in G \tag{19}$$

$$x_{ijk} = 0 \qquad \forall i \in D, j \neq s(i), k \in G \tag{20}$$

$$0 \leqslant y_{ij} \leqslant 1 \qquad \forall i \in S \setminus D, j \in Z_i \tag{21}$$

Observe that if $D = \varnothing$, the resulting model is the LP-relaxation of TQDF, whereas if $D = S$, we arrive at the situation where an interval has been selected for each supplier (see [7]).

Theorem 5 *GENTQDF can be polynomially transformed to min-cost flow.*

PROOF. We organize the proof by first showing that an optimal solution of GENTQDF has a structural property. Then we construct a min-cost flow instance and show the correspondence between optimal solutions of this instance and GENTQDF.

Claim: There exists an optimal solution (x^*, y^*) of GENTQDF in which for each $i \in S \setminus D$:

$$\begin{array}{ll} x^*_{ijk} = 0 & \forall j \neq max_i, \forall k \in G, \text{ and} \\ y^*_{ij} = 0 & \forall j \neq max_i. \end{array} \tag{22}$$

Thus, the claim states that there exists an optimal solution in which all x- and y-variables equal 0, except those corresponding to the highest interval of each supplier. In other words, goods are bought only at the lowest prices of each

supplier.

Argument: given some feasible solution (x, y) of GENTQDF, we show how to modify (x, y) to (x', y') such that (x', y') is a feasible solution of GENTQDF satisfying (22) and such that the cost of (x', y') does not exceed the cost of (x, y).

For each $k \in G$ and each $i \in S \setminus D$, we set

$$x'_{i,max_i,k} = \sum_{j=0}^{max_i} x_{i,j,k}, \text{ and} \tag{23}$$

$$x'_{i,j,k} = 0 \qquad \text{for } j = 0, 1, ..., max_i - 1. \tag{24}$$

Further, for each $i \in S \setminus D$, we set

$$y'_{i,max_i} = y_{i,max_i} + \frac{\sum_{j=0}^{max_i-1} \sum_{k \in G} x_{i,j,k}}{u_{i,max_i}}, \text{ and} \tag{25}$$

$$y'_{i,j} = 0 \qquad \text{for } j = 0, 1, ..., max_i - 1. \tag{26}$$

All other variables remain the same, that is

$$x'_{i,j,k} = x_{i,j,k} \qquad \forall i \in D, j \in Z_i, k \in G. \tag{27}$$

It is obvious that the costs of (x', y') cannot exceed the costs of (x, y) since the total amount of goods has remained the same for each supplier, while in (x', y') all goods are purchased in the highest interval (and we have $c_{i,max_i,k} \leqslant c_{i,j,k}$ $\forall i, j, k$, see (1)). Let us now argue that (x', y') is a feasible solution of GENTQDF.

Evidently, (x', y') satisfies (13), (17), (18), (19) and (20). To show that (x', y') satisfies (14) and (21), we need to show that $y'_{i,max_i} \leqslant 1$ for $i \in S \setminus D$. Observe that for $j = 0, 1, ..., max_i - 1$ we have $\sum_{k \in G} x_{ijk}/u_{ij} \leqslant y_{ij}$ (using the feasibility of (x, y) with respect to (16)) and thus $\sum_{k \in G} x_{ijk}/u_{i,max_i} \leqslant y_{ij}$ for $j = 0, 1, ..., max_i - 1$. Summing over $j = 0, 1, ..., max_i - 1$ implies that $\sum_{j=0}^{max_i-1} (\sum_{k \in G} x_{ijk})/u_{i,max_i} \leqslant \sum_{j=0}^{max_i-1} y_{ij}$ and together with the feasibility of (x, y) with respect to (14) this leads to (x', y') satisfying (14) and (21).

Consider now for some $i \in S \setminus D$ constraints (15), written alternatively as $\sum_{k \in G} x_{ijk} \geqslant l_{ij} y_{ij}$ for $j = 0, 1, ..., max_i$. In case $j < max_i$, the right-hand side equals 0 (since $y'_{ij} = 0$ for $j < max_i$ by construction) and feasibility follows. In case $j = max_i$, we have, using feasibility of (x, y), that

$$\sum_{k \in G} x_{i,max_i,k} \geqslant l_{i,max_i} y_{i,max_i}. \tag{28}$$

Also it is true that

$$\sum_{j=0}^{max_i-1} \sum_{k \in G} x_{ijk} \geqslant \frac{\sum_{j=0}^{max_i-1} \sum_{k \in G} x_{ijk}}{u_{i,max_i}} l_{i,max_i}. \tag{29}$$

Summing (28) and (29) yields:

$$\sum_{k\in G} x'_{i,max_i,k} = \sum_{k\in G}(x_{i,max_i,k} + \sum_{j=0}^{max_i-1} x_{i,j,k})$$
$$\geqslant l_{i,max_i}(y_{i,max_i} + \sum_{j=0}^{max_i-1}\sum_{k\in G}\frac{x_{i,j,k}}{u_{i,max_i}}) = l_{i,max_i}y'_{i,max_i}. \quad (30)$$

Thus (x', y') satisfies constraints (15).

To verify that (x', y') satisfies constraints (16), observe that for $i \in S\setminus D$ and for $j = 0, 1, ..., max_i - 1$ $\sum_{k\in G} x'_{i,j,k} = 0$ and $y'_{ij} = 0$ (this follows by construction of x' and y'). Finally, in case $j = max_i$ we have

$$\sum_{k\in G} x_{i,max_i,k} \leqslant u_{i,max_i}y_{i,max_i}, \text{ and} \quad (31)$$
$$\sum_{j=0}^{max_i-1}\sum_{k\in G} x_{ijk} = \frac{\sum_{j=0}^{max_i-1}\sum_{k\in G} x_{ijk}}{u_{i,max_i}}u_{i,max_i}. \quad (32)$$

Summing (31) and (32) yields

$$\sum_{k\in G} x'_{i,max_i,k} = \sum_{k\in G}(x_{i,max_i,k} + \sum_{j=0}^{max_i-1} x_{i,j,k})$$
$$\leqslant u_{i,max_i}(y_{i,max_i} + \sum_{j=0}^{max_i-1}\sum_{k\in G}\frac{x_{i,j,k}}{u_{i,max_i}}) = u_{i,max_i}y'_{i,max_i}, \quad (33)$$

which shows that constraints (16) are also satisfied by (x', y') and allows us to conclude that (x', y') is indeed a feasible solution of GENTQDF.

Let us now build the network. We have three sets of nodes: there is a node for each supplier (a 'supplier node'), there is a node for each good (a 'good node') and there is a single source node. The supply of the source node equals $\sum_{k\in G} d_k$ and the demand of each good node equals d_k. All other demands are 0. Furthermore, there is an arc from the source node to each supplier node. If this supplier is in D, the corresponding lower and upper bounds of this arc are $l_{i,s(i)}$ and $u_{i,s(i)}$; if this supplier is not in D, the lower and upper bounds are 0 and u_{i,max_i}. (The choice for a lower bound of 0 for suppliers not in D, even if $l_{i,0}$ is strictly positive, may seem surprising at first sight. It can however be verified that because the y-values are relaxed in GENTQDF, $l_{i,0}$ no longer constrains the x-values.) The cost of an arc between the source node and each supplier node equals 0. There are also arcs from each supplier node to each good node. These arcs are not constrained by lower or upper bounds, but do have a cost equal to $c_{i,s(i),k}$ if the corresponding supplier is in D and equal to $c_{i,max_i,k}$ if this supplier is not in D. This completes the description of the min-cost flow instance. A schematic representation is given in Figure 1.

A solution of this min-cost flow instance is characterized by flows $f_{i,k}$ on each arc from supplier i to good k. It corresponds to a solution of GENTQDF as follows:

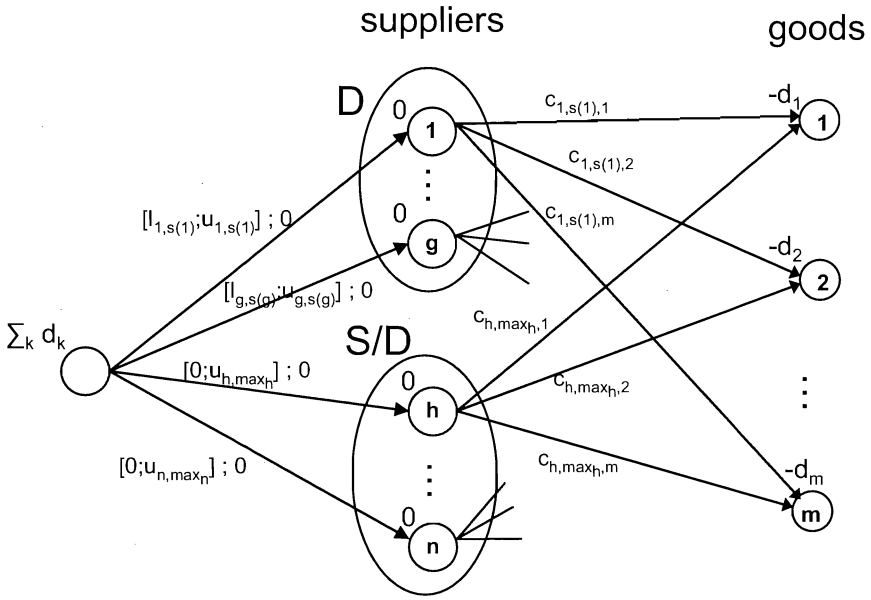

Figure 1: GENTQDF as min-cost flow

$$x_{i,s(i),k} = f_{i,k} \qquad \forall i \in D, k \in G, \tag{34}$$

$$x_{i,max_i,k} = f_{i,k} \qquad \forall i \notin D, k \in G, \tag{35}$$

$$y_{i,s(i)} = \sum_{k \in G} \frac{f_{i,k}}{u_{i,s(i)}} \qquad \forall i \in D, \tag{36}$$

$$y_{i,max_i} = \sum_{k \in G} \frac{f_{i,k}}{u_{i,max_i}} \qquad \forall i \notin D. \tag{37}$$

All other x- and y-variables of GENTQDF are set equal to 0.

Given (22), we conclude that an optimal solution of the min-cost flow problem in Figure 1 corresponds to an optimal solution of GENTQDF. It can now easily be seen that an optimal solution of GENTQDF also corresponds to an optimal flow in the min-cost flow problem. Thus, we have shown how GENTQDF can be polynomially transformed to min-cost flow. □

Notice that as a consequence of Theorem 5, the LP-relaxation of TQDF can be found by solving a min-cost flow problem. This result is the foundation for an exact algorithm to be discussed in section 6.

4 Variants of the TQD problem

When procuring goods, other considerations besides the price can be relevant. Although our model does not incorporate criteria like quality or reliability, we

now consider a number of variants of the TQD problem that are common in both practice and literature. A first variant adds constraints on the amount of goods the buyer is willing to purchase from a supplier. In another variant, the buyer is allowed to buy more goods than strictly needed, while the third variant imposes a restriction on the number of winning suppliers (suppliers that end up selling some amount of any of the goods are called winning suppliers). We show that results similar to that of Theorem 5 hold for each of these variants.

4.1 Market share constraints

Suppose that the buyer wants to impose upper and/or lower bounds on the amount of a good that must be ordered from a supplier. Forcing that some supplier i must be allocated an amount of at least $q_{i,k}$ and at most $Q_{i,k}$ of good k can be done by adding the following constraint to GENTQDF:

$$q_{i,k} \leqslant \sum_{j \in Z_i} x_{ijk} \leqslant Q_{i,k}. \tag{38}$$

On a more global level the buyer could provide bounds on the total allocation for a supplier, across all goods. Forcing the total amount of goods purchased from a supplier i to lie between w_i and W_i can be done by adding the following constraint to GENTQDF:

$$w_i \leqslant \sum_{j \in Z_i} \sum_{k \in G} x_{ijk} \leqslant W_i. \tag{39}$$

These market share constraints are often mentioned in literature (see [3], [4], [5], and [6]). Notice that none of these extra constraints invalidate property (22). Constraints (38) can easily be implemented in the min-cost flow graph by changing the lower and upper bounds of the arcs from supplier i to good k. Constraints (39) can be realized via the lower and upper bounds of the arcs from the root node to supplier i. Thus, we obtain the following statement:

Theorem 6 *GENTQDF with constraints (38) and/or (39) can be polynomially transformed to min-cost flow.*

4.2 More-for-less

As described in section 3.1, it can be advantageous to obtain more of some good k than the required amount d_k, since this might allow the buyer to use the cheaper prices of a higher interval (see also [2] and [10]). If we wish to allow this, constraints (13) in GENTQDF should be replaced by

$$\sum_{i \in S} \sum_{j \in Z_i} x_{ijk} \geqslant d_k \qquad \forall k \in G. \tag{40}$$

Notice that for the special case where $D = \emptyset$, all units are already bought in the highest intervals in an optimal solution of GENTQDF (see (22)). Therefore, there is no need to buy more than d_k of any good k and an optimal solution can be found by solving the min-cost flow problem in Figure 1. In general however, we can formulate the following result:

Theorem 7 *GENTQDF with constraints (13) replaced by (40) can be polynomially transformed to min-cost flow.*

PROOF. Consider the graph in Figure 2. It has supplier and good nodes, with demands and connecting arcs like in Figure 1. The lower and upper bounds and the costs for these arcs are the same as in Figure 1 but in order not to overload the figure, they have been omitted. There is however also a dummy node, corresponding to the additional goods that are bought once the demand d_k is fulfilled. The dummy node has a demand of M, being a large number. The supply of the source node is increased by this same amount M. Furthermore, there is an arc from the source node to the dummy node with cost 0 and an upper bound of M. Notice that any flow in the network in Figure 1 is still a feasible flow in the network in Figure 2. There are also arcs from each supplier $i \in D$ to the dummy node. These arcs have a cost equal to the price of the supplier's cheapest good in its selected interval $s(i)$. In Figure 2, we refer to this good as $q(i)$. Notice that this is the good we will buy additionally from that supplier to reach the threshold of a higher interval; it would be pointless to buy a more expensive good instead to achieve this. There are no arcs to the dummy node from suppliers not in D. Since for these suppliers the goods are already bought at their lowest prices (see (22)), there is no use in buying additional goods. □

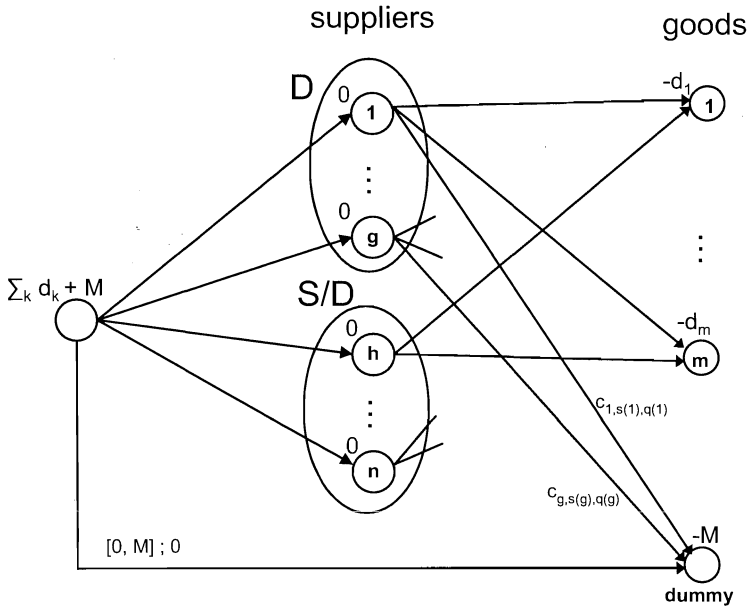

Figure 2: GENTQDF with more-for-less as min-cost flow

Observe that in GENTQDF it can happen that because of the interval selections made for suppliers in D, no feasible solution exists. This is the case if the demands d_k are not high enough to reach the required lower bounds of the selected intervals. In the more-for-less variant of GENTQDF however, this is no longer possible since it is allowed to buy more than the amounts d_k. Indeed,

these extra amounts correspond to the flows on the arcs from suppliers in D to the dummy node.

4.3 Limited number of winning suppliers

Another important consideration apart from cost minimization is to make sure that the demand is not procured from too many suppliers (see also [3], [4], [5], [6] and [10]). Otherwise, overhead costs increase due to managing this large amount of suppliers.

In order to model the requirement that a limited number of suppliers is selected, we need to understand exactly when a supplier receives a positive amount. This happens when $y_{ij} = 1$ for some j, except possibly when $j = 0$, and $l_{i0} = 0$; the latter situation refers to the case where interval 0, with a lower bound of 0, is selected. Then a supplier might receive nothing, while there is a y-variable with a positive value. To handle this situation, we 'split' each interval that has a lower bound of 0 and a positive upper bound into two intervals: one interval with a lower bound and an upper bound of 0 (the dummy interval), and one interval with a lower bound of 1 and an upper bound equal to the original upper bound (interval 0). Thus, we have redefined interval 0 by excluding the option of a zero amount of goods. Moreover, we let y_{i0} correspond to this new interval 0. Obviously, selecting a supplier's dummy interval comes down to not selecting this supplier at all, in which case the supplier can simply be removed from the problem. Selecting another interval of a supplier implies that this is a winning supplier. This approach leads to a set D, containing only winning suppliers. In fact, without loss of generality, we can now focus on constraining the winning suppliers not in D, and limit their number to K by adding the following constraint to GENTQDF:

$$\sum_{i \in S \setminus D} \sum_{j \in Z_i} y_{ij} \leqslant K. \tag{41}$$

If we assume that the highest volume interval of every supplier in $S \setminus D$ has the same upper bound, we can prove a similar result to that of Theorem 5. We refer to this common upper bound as u_{max}. Given the fact that in most real-life applications suppliers pose no upper bound at all to the amount of goods they are willing to sell, this assumption is quite reasonable.

Theorem 8 *If $u_{max_i} = u_{max}$ $\forall i \in S \setminus D$, then GENTQDF with constraint (41) added can be polynomially transformed to a min-cost flow problem.*

PROOF. First, notice that property (22) remains valid in this setting. Indeed, given the x-values, we can find y-values for each supplier $i \in S \setminus D$ and each volume interval $j \in Z_i$ satisfying constraints (15) and (16) in the following interval:

$$[\frac{\sum_{k \in G} x_{ijk}}{u_{i,j}}, \frac{\sum_{k \in G} x_{ijk}}{l_{i,j}}]. \tag{42}$$

Naturally, in order to fulfill constraints (21), the y-values cannot exceed 1. It is easy to verify that shifting goods from a supplier's highest interval to one

or more lower intervals can never decrease the total y-value of this supplier. Therefore, constraint (41) will never force the optimal solution of GENTQDF away from the highest intervals and property (22) still holds.

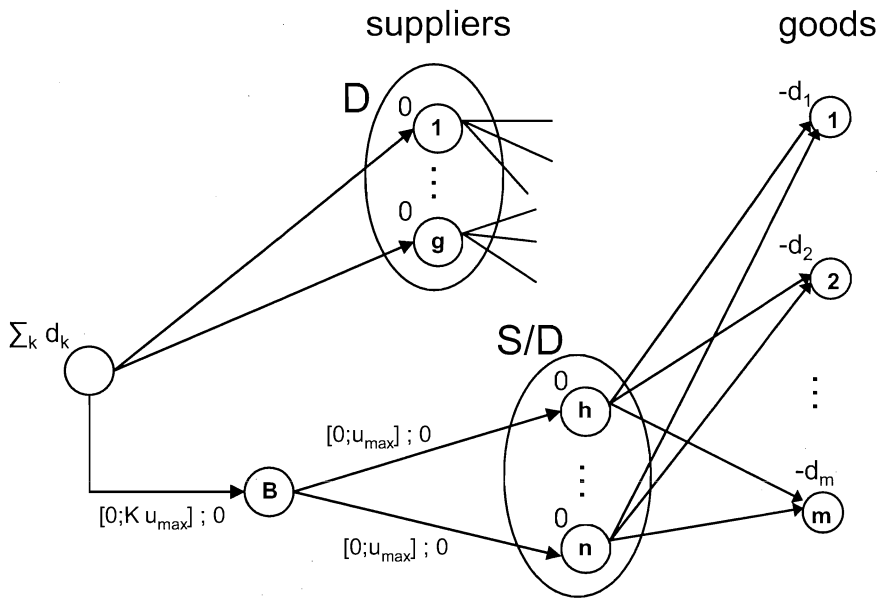

Figure 3: GENTQDF with a limited number of winning suppliers as min-cost flow

We can now construct a min-cost flow network (see Figure 3). Compared to Figure 1, an extra node, referred to as B, is added. The arc from the root node to B has an upper bound of Ku_{max}, and the arcs from B to the supplier nodes have upper bounds of u_{max}.

Let d_{min} be the minimal amount of goods that needs to be purchased from suppliers not in D in order to have a feasible solution, i.e., $d_{min} = \sum_{k \in G} d_k - \sum_{i \in D} u_{i,s(i)}$. The min-cost flow problem can only be infeasible if this demand d_{min} is too high for the upper bounds on the arcs, i.e., if $d_{min} > Ku_{max}$. In this case however, GENTQDF with constraint (41) is infeasible as well. Indeed, even when choosing the y-values as low as possible, namely as f_i/u_{max}, we fail to meet constraint (41):

$$\begin{aligned} \sum_{i \in S \setminus D} \sum_{j \in Z_i} y_{ij} &= \sum_{i \in S \setminus D} f_i/u_{max} \\ &\geqslant d_{min}/u_{max} \\ &> K. \end{aligned}$$

If there exists a feasible flow f to the min-cost flow problem, then we can always find a solution to GENTQDF with constraint (41) by setting the x- and y-variables as in (34)-(37). From Theorem 5, it is clear that this solution satisfies

(13)-(21). Let d_{max} be the maximal amount of goods that can be purchased from suppliers not in D in order to keep the solution feasible, i.e., $d_{max} = \sum_{k \in G} d_k - \sum_{i \in D} l_{i,s(i)}$. Obviously, a feasible flow will have $d_{max} \leqslant K u_{max}$. Therefore, the resulting y-variables will also satisfy (41), as shown below:

$$\begin{aligned} \sum_{i \in S \setminus D} \sum_{j \in Z_i} y_{ij} &= \sum_{i \in S \setminus D} \sum_{k \in G} f_{i,k}/u_{max} \\ &\leqslant \sum_{k} d_{max}/u_{max} \\ &\leqslant K. \end{aligned}$$

□

Property (22) is crucial for the possibility to use min-cost flow to solve LP-relaxations of GENTQDF-type formulations. For instance, one could also argue that the number of winning suppliers must be at least a minimum number, say L. Indeed, depending on too few suppliers could move the buyer in a vulnerable position if one of these suppliers is unable to supply as agreed. This could be encoded by adding the following constraint to GENTQDF:

$$\sum_{i \in S} \sum_{j \in Z_i} y_{ij} \geqslant L. \tag{43}$$

The min-cost flow approach however does not apply to this setting. Constraint (43) pushes the optimal solution away from the highest intervals, since moving the goods towards one or more lower intervals can increase the total y-value of each supplier. It is easily verified that therefore property (22) is no longer valid, which prevents us from following the same reasoning as in Theorem 5.

5 Exact algorithms

In this section we describe the three exact algorithms used to solve instances of the TQD problem and its variants. First, we explain the min-cost flow based branch-and-bound algorithm. We build a branching tree such that in every node a min-cost flow problem needs to be solved (see Theorem 5). The branching tree is constructed in such a way that every level in the tree corresponds to a supplier, and that there is a branch for every volume interval of that supplier.

In the root node, the LP relaxation of the TQD problem is solved as explained in section 3.2. The x-values in the root node solution point us to an interval for each supplier. For each supplier, we compute its priority as the number of volume intervals minus the index of the interval suggested by the root node solution. Thus, suppliers that announce a lot of volume intervals but receive little in the LP-relaxation, are accorded the highest priority. We use this priority to build up the search tree, as we start with the supplier with the highest priority, creating branches from the root node for each of its intervals. In the node from the first branch, we fix the volume interval in which we end up according to

the values of the x-variables in the LP relaxation. In the next branch of that level, we fix the interval directly above this interval; in the following branch and still within this level, we fix the interval directly below it and so on (provided that these intervals exist). In the following level of the branching tree we continue with the supplier with the second highest priority, again branching on its intervals as just explained, and so on. Naturally, there is no need to create a node in the branching tree for a supplier with only one interval, since we can fix this interval right away. To traverse the tree, we use a standard depth-first search strategy where, as usual, a node is fathomed if its solution is dominated by the current best solution or if it is infeasible. We experimented with different priority settings, and the choice described above seems to work best.

The branching tree for both the market share and the more-for-less variant is very similar. In the first variant, we prune the tree by deleting those volume intervals that fall outside the range imposed by the market-share constraints. Afterwards, we can adapt the upper and lower bounds of the highest and lowest interval respectively according to the market share constraints. As a result, the branching tree is typically sparser in the market share variant than in the basic case. In the more-for-less variant on the other hand, the branching tree is in general dense compared to its counterpart in the basic case, because less nodes are infeasible in the more-for-less setting.

The branching tree for the variant that limits the number of winning suppliers to K differs from the branching tree of the basic case, because we need to introduce an extra branch on every level of the tree. This branch corresponds to the dummy interval as introduced in section 4.3 and imposes that the corresponding supplier is not to be used in the solution. Whereas suppliers with only one interval are left out of the tree completely in the basic case, they now appear in the tree with two branches, representing the decision to buy from that supplier or not. On the other hand, a node needs no further branching as soon as K suppliers have been selected. For all three variants, we use the same depth-first strategy as for the basic case.

The min-cost flow based branch-and-bound algorithm has been programmed in C and compiled using Microsoft Visual C++ 6.0. To solve the min-cost flow problems, we have used the network solver of Ilog Cplex 8.1.

The description of the other two algorithms is straightforward. The branch-and-cut algorithm simply uses the default settings of the MIP solver of Ilog Cplex 8.1. To study the effect of the cuts, we have also investigated another algorithm in which we disallow the Ilog Cplex MIP solver to generate cuts. We refer to this algorithm as the linear programming based branch-and-bound algorithm. This algorithm uses a best-bound node-selection strategy instead of a depth-first search, but more importantly, it uses the shadow prices of the y-variables to select the branching variable at the node which has been selected for branching.

6 Computational results

In this section we discuss the choices that were made to construct the instances on which the algorithms have been tested. We continue with computational results for the TQD problem and its variants and evaluate the performance of our algorithms.

6.1 Structure of the instances

In order to test the performance of the exact algorithms, two types of instances have been generated: instances with a special structure and completely random instances. All instances have 10, 20 or 50 suppliers and 40 or 100 goods. Furthermore, each supplier has a maximum of 3 or 5 volume intervals. For all instances, the total demand for a good is a random number between 1000 and 10000. For instances with 40 goods, the upperbound-increase from one interval to the next is a random number between 10000 and 50000, while for instances with 100 goods, the upperbound-increase is a random number between 10000 and 100000.

For structured instances, we first determine a base price for each good, randomly picked between 3 and 7. The price for a good in a supplier's first interval is then computed by adding a random number in the interval $[-2, 2]$ to the base price. Furthermore, for each supplier i there is a discount rate $\delta_{ij} \in (0, 0.1)$ for every interval $j > 1$, which determines the price $c_{i,j,k}$ of good k in interval j as a function of the price in interval $j-1$ as follows:

$$c_{i,j,k} = (1 - \delta_{ij}) c_{i,j-1,k} \qquad \forall i, k \text{ and } \forall j > 1 \tag{44}$$

For random instances, the cost of purchasing a good from a supplier in its first interval is a random number between 2 and 8. The price for this good in each of the next intervals is computed by discounting the price in the previous interval by a percentage picked randomly between 0 and 75%.

The key difference between the random and the structured instances is that for the former instances prices can drop drastically from one interval to the next, whereas for the latter this decrease in price is limited to 10%. Furthermore, for the structured instances, a good that is expensive at one supplier will very likely be expensive at the other suppliers too. For the random instances however, this is not necessarily the case as prices for a good can differ in a wider range between the various suppliers. Finally, the discount percentage one receives when moving from one interval to the next can differ substantially between the goods for the random instances, while it is the same for all the goods for the structured instances.

In the variant with the market share constraints, only global constraints (as in (39)) are included. For the instances with 10 suppliers, 5 suppliers are picked randomly from each of whom between 5 and 20 percent of the total demand needs to be purchased. For instances with 20 suppliers, we pick 10 suppliers

and force between 5 and 15 % of the total demand to go to each of them and for the instances with 50 suppliers this becomes 20 suppliers with each 5 to 10 % of the total demand. The more-for-less variant needs no extra modifications, apart from allowing to buy more than what is demanded. For the third variant, the number of winning suppliers is limited to 5 for all instances. If an instance has no solution with only 5 winning suppliers, the interval thresholds are doubled for each supplier until a solution exists.

6.2 Results

The results of our experiments are summarized in tables 1 to 4. The instances are coded with 'S' for structured and 'R' for random instances. The first number indicates the number of suppliers, the second number reflects the number of goods and the third number is the maximal number of intervals per supplier. For each of these types of instances, 10 instances were generated and solved with the three algorithms. This resulted in computation times (in seconds) and a number of nodes searched in the branching tree for each algorithm, averaged per type of instance in the table. All computations were done on a Pentium *IV* 2 GHz computer, with 512 Mb RAM.

In Table 1, the results for the basic TQD problem are presented. Each algorithm solves all instances in a reasonable amount of time; random instances seem to be harder to solve than the structured ones. The min-cost flow based algorithm clearly performs best in terms of computation time for all instances with 10 or 20 suppliers. However, instances with 50 suppliers prove to be harder to solve with this algorithm. Although the solution time per node is undoubtedly the smallest with the min-cost flow approach, it needs more computing time than the other two exact algorithms. The branch-and-cut approach clearly searches the least amount of nodes, but to achieve this it needs a time-consuming cut generation process. The results show that it pays to generate cuts when the number of suppliers is large.

The results of our experiments with the variant with market share constraints are summarized in Table 2. As in the basic case, the random instances require more computation time than the structured ones. Market share constraints are problematic for the branch-and-cut algorithm, whose computation times sometimes even double compared to the basic case. The linear programming based branch-and-bound algorithm deals with these constraints much better, since it manages to solve the instances faster than in the basic case. The min-cost flow algorithm is however by far the fastest algorithm for all instances. Especially for the instances with 50 suppliers, adding market share constraints causes the computations times to slump compared to the basic case. Moreover, less nodes need to be searched, which can be explained by the construction of the branching tree as described in section 5.

Table 3 figures the results for the more-for-less variant. It turns out that in none of the structured instances purchasing extra goods leads to a lower total cost. In the random instances however, it is profitable in more than 85% of the instances to buy more than strictly needed. This is explained by the fact that

discounts are substantially larger for the random cases than for the structured instances (see section 6.1).

Once again, the min-cost flow based algorithm performs best on all instances with 10 or 20 suppliers. For instances with 50 suppliers, it is advisable to use the linear programming based branch-and-bound algorithm. Compared to the basic case, the min-cost flow algorithm needs to search slightly more nodes, resulting in more computation time. Apart from the random instances with 50 suppliers, which seem very difficult for all algorithms, this increase in computation time remains very modest. The linear programming based branch-and-bound algorithm is not affected too much either. The branch-and-cut algorithm however deals poorly with this variant.

Finally, Table 4 describes the results for the variant that limits the number of winning suppliers. This constraint proved to be binding for more than 98% of the structured instances but less than 50% of the random instances. For the random instances, the prices drop sharper from one interval to the next, which makes it more interesting to go for the higher intervals. This leads to an optimal solution with less suppliers than for the structured instances. This explains why a constraint limiting the number of winning suppliers less often affects the random instances.

As for the computation times, branch-and-cut seems the best option for the structured instances. For the random instances, the picture is less clear. The instances with 10 suppliers are best solved with the min-cost flow algorithm, although this algorithm is left far behind by the other two for the instances with 20 suppliers. For these instances, branch-and-bound based on linear programming outperforms the other algorithms for instances where suppliers can have up to 5 volume intervals. Branch-and-cut is the fastest approach to solve random instances with 20 suppliers and up to 3 volume intervals per supplier. Notice that no instances with 50 suppliers are mentioned in this table, because the computation times for these problems were impractically high for all algorithms.

7 Conclusions

We presented a procurement problem where suppliers adopt a discount that depends on the total quantity ordered. We argued that different versions of this problem are NP-hard and that it is impossible to find a polynomial-time approximation algorithm with a constant ratio (unless $P = NP$). We described three exact algorithms: one algorithm is based on our result that the problem can be solved by solving a number of min-cost flow problems; the other two algorithms are a branch-and-cut and an linear programming based branch-and-bound algorithm.

The algorithms were tested on fairly large randomly generated instances of the basic problem and three variants. Our computational results show that all three algorithms came to an exact solution in a reasonable amount of time. However,

it also became clear that each algorithm has instances for which it performs best. In general, the min-cost flow based algorithm works best for small instances in terms of number of suppliers. It works especially well for the variant where we imposed constraints on the market share a supplier is allowed to obtain. The branch-and-cut algorithm outperforms the other algorithms on large instances in terms of suppliers of the basic case and on the structured instances of the variant that requires a limited amount of winning suppliers. Finally, the linear programming based branch-and-bound algorithm is at its best with the large instances of the variant where the buyer is allowed to purchase more than strictly needed.

Appendix

Theorem 1 *The decision version of the TQD problem is strongly NP-complete.*

PROOF. We define TQD' as the decision version of the TQD problem, where the question is whether it is possible to buy the required goods at a given total purchasing cost K. Obviously, TQD' is in NP, since given a solution it suffices to check the constraints and the value of the solution, which can easily be done in polynomial time. The reduction is from the 3-dimensional matching (3DM) problem.

The decision version of the 3DM problem is described as follows: given a set $M \subseteq X \times Y \times Z$ of triples, where each of the sets X, Y and Z has exactly q elements, is there a matching in M that contains q triples? Every instance of 3DM can be reduced to a TQD' instance in polynomial time. Suppose that the $3q$ elements of the sets X, Y, and Z correspond to $3q$ goods and that each 3-element subset in M corresponds to a supplier, so $n = q$ and $m = 3q$. Each supplier has 2 intervals. The price of each good in its first interval is 1. This interval has a lower bound of 0 and an upper bound of 2. The second interval has a lower bound of 3 and an upper bound of ∞. The price of each good in this second interval is also 1, except for the three goods in the 3-element subset corresponding to the supplier, each of which have a price of 0. Each good needs to be purchased exactly once, i.e., $d_k = 1\ \forall k$. The question is whether the TQD' problem can be solved with a total purchasing cost of 0.

Further, every yes-instance of 3DM corresponds to a yes-instance of TQD'. A solution of 3DM consists of q 3-element subsets, corresponding to q suppliers in the TQD' problem. Purchasing from each of these suppliers exactly the 3 goods represented by the 3-element subset enables us to reach every supplier's second interval, where these 3 goods can be bought at price 0. Since every element of $X \cup Y \cup Z$ occurs exactly once in the solution of 3DM, every good will also be purchased exactly once in the TQD' solution. Therefore, if 3DM has a solution, it can easily be transformed to a solution of TQD'.

Vice versa, every yes-instance of TQD' also corresponds to a yes-instance of 3DM. A solution of the TQD' problem consists of a number of selected suppliers, together providing every good exactly once at a total cost of 0. If a supplier would provide less than 3 goods, the quorum to get in the second interval would not be met, so the cost would not be 0. If the supplier would provide more, the cost would also be strictly positive, because all but these 3 goods still have a price of 1 in the second interval. Providing more than one of the 0-priced goods would violate the demand constraint stating that each good is to be supplied exactly once. Therefore every selected supplier provides precisely 3 goods, namely those that have a price of 0 in the second interval and since $3q$ goods need to be provided, q suppliers must be selected. Therefore, for each of the q suppliers selected in the solution of the TQD' problem, there is a corresponding 3-element set in M. Moreover, these q triples define a matching, since every good is bought exactly once. As a consequence, the decision version of the TQD problem is strongly NP-complete. □

Theorem 2 *No polynomial-time approximation algorithm with constant worst-case ratio exists for the TQD problem (unless $P = NP$).*

PROOF. Assume that a ρ-approximation algorithm for the TQD problem exists. Consider now an instance of 3DM with $M \subseteq X \times Y \times Z$, and let us build an instance of the TQD problem as in the proof of Theorem 1 with a price of $\rho+1$ for any good bought in the first interval, or bought in the second interval when not belonging to one of the three goods of that supplier. Observe that this instance of the TQD problem either has an optimal solution with cost 0 (namely when the 3DM-instance has a matching), or it has an optimal solution with cost at least $\rho+1$ (when there is no matching in the 3DM instance). Thus, if there is a 3DM-matching the ρ-approximation algorithm must return a zero-cost solution, which contradicts the NP-hardness of 3DM. Hence such an algorithm cannot exist unless $P = NP$. □

Theorem 3 *The decision version of the TQD problem with a common discount rate δ is strongly NP-complete.*

PROOF. In order to show that the TQD problem with a common discount rate is NP-complete, we modify the reduction used in Theorem 1 as follows. As in Theorem 1, each supplier has 2 intervals, the first interval ranges from 0 to 2 goods, the second from 3 to an unlimited amount of goods. The prices of all goods in both the first and second interval are 1, except for the three goods in the 3-element subset corresponding to the supplier, each of which have a price of $(1 - \delta)$. Each good still needs to be purchased exactly once. The question is now whether this TQD problem can be solved with a total purchasing cost of $m(1 - \delta)$. The same reasoning as in Theorem 1 can be applied to verify that every yes-instance of 3DM corresponds to a yes-instance of the TQD problem with common discount rate and vice versa and that indeed the decision version of the TQD problem with a common discount rate is strongly NP-complete. □

Theorem 4 *The decision version of the more-for-less variant of TQD problem is strongly NP-complete.*

PROOF. In the more-for-less setting, the buyer is allowed to purchase more than m goods in order to reduce the total cost. We can however use the same reduction as in Theorem 3. Indeed, let each supplier have 2 intervals, the first ranging from 0 to 2 goods, the second from 3 to an unlimited amount of goods. Once again, the prices of all goods in both the first and second interval are 1, except for the three goods in the 3-element subset corresponding to the supplier, each of which have a price of $(1-\delta)$. The question remains whether it is possible to solve this TQD problem with a total purchasing cost of $m(1 - \delta)$. Clearly this can not be achieved by purchasing more than m goods, which allows us to conclude that every yes-instance of 3DM corresponds to a yes-instance of the more-for-less variant and vice versa. Hence, the decision version of the more-for-less variant of the TQD problem is strongly NP-complete. □

References

[1] M. Bichler, J.R. Kalagnanam, H.S. Lee, J. Lee. (2002) *Winner Determination Algorithms for Electronic Auctions: A Framework Design.* In: Proceedings of EC-Web 2002, pp. 37-46.

[2] Y. Crama, R. Pascual J. and A. Torres (2004). *Optimal procurement decisions in the presence of total quantity discounts and alternative product recipes.* European Journal of Operational Research, 159(2) pp. 364-378.

[3] A.J. Davenport and J.R. Kalagnanam (2002). *Price negotiations for procurement of direct inputs.* In: B. Dietrich and R.V. Vohra, Mathematics of the internet: e-auction and markets, pp. 27-43.

[4] M. Eso, S. Ghosh, J.R. Kalagnanam and L. Ladanyi (2001). *Bid evaluation in procurement auctions with piece-wise linear supply curves.* IBM Research Report RC 22219, 2001.

[5] G. Hohner, J. Rich, E. Ng, G. Reid, A.J. Davenport, J.R. Kalagnanam, H.S. Lee, and C. An (2003). *Combinatorial and Quantity-Discount Procurement Auctions Benefit Mars, Incorporated and Its Suppliers.* Interfaces, 33(1) pp. 23-35.

[6] P. Katz, A.A. Sadrian and P. Tendick (1994). *Telephone Companies Analyze Price Quotations with Bellcore's PDSS Software.* Interfaces, 24(1) pp. 50-63.

[7] J.J. van de Klundert, J. Kuipers, F.C.R. Spieksma and M. Winkels (2003). *Telecommunication Carrier Selection under Volume Discounts: a Case Study.* Research Report 0330, Department of Applied Economics, K.U.Leuven, 2003, accepted for Interfaces.

[8] A. Kothari, D. Parkes and S. Suri (2003). *Approximately-strategyproof and tractable multi-unit auctions.* In: Proceedings of the ACM Conference on Electronic Commerce 2003, pp. 166-175.

[9] C.L. Munson and M.J. Rosenblatt (1998). *Theories and realities of quantity discounts: an explanatory study.* Production and Operations Management, 7(4) pp. 352-369.

[10] A.A. Sadrian and Y.S. Yoon (1994). *A Procurement Decision Support System in Business Volume Discount Environments.* Operations Research, 42(1) pp. 14-23.

[11] H. Shachnai, O. Shmueli and R. Sayegh (2004). *Approximation Schemes for Deal Splitting and Covering Integer Programs with Multiplicity Constraints.* Manuscript.

[12] J. Xu, L.L. Lu and F. Glover (2000). *The deterministic multi-item dynamic lot size problem with joint business volume discount.* Annals of Operations Research, 96(1) pp. 317-337

Instances	mcf branch&bound		branch&cut		lp branch&bound	
	comp. time	#nodes	comp. time	#nodes	comp. time	#nodes
S-10-40-3	0,01	116,6	0,09	0,3	0,08	29,3
S-10-40-5	0,02	161,1	0,15	10,9	0,11	52,5
S-10-100-3	0,02	69,8	0,12	0,2	0,11	17,2
S-10-100-5	0,14	501,6	0,55	3,2	0,36	74,3
S-20-40-3	0,07	389,2	0,12	0,5	0,16	73,5
S-20-40-5	0,38	1.887,8	0,50	4,7	0,58	207,3
S-20-100-3	0,30	749,6	0,34	1,3	0,57	128,8
S-20-100-5	0,67	1.512,8	1,17	2,1	1,07	155,1
S-50-40-3	5,61	16.671,8	0,51	2,7	1,92	719,0
S-50-40-5	32,93	85.210,4	2,99	16,5	7,81	2.087,2
S-50-100-3	21,81	26.595,3	1,45	2,1	4,67	696,1
S-50-100-5	159,77	168.181,3	10,45	14,7	24,41	2.614,0
R-10-40-3	0,01	54,9	0,09	2,1	0,07	24,3
R-10-40-5	0,07	428,9	0,59	30,5	0,31	160,7
R-10-100-3	0,02	46,9	0,14	2,6	0,10	10,1
R-10-100-5	0,31	845,1	1,50	31,7	0,78	160,0
R-20-40-3	0,14	700,5	0,29	9,5	0,25	121,6
R-20-40-5	0,45	2.155,6	1,81	68,5	1,56	659,9
R-20-100-3	0,59	1.249,6	0,83	8,1	1,05	235,0
R-20-100-5	3,18	3.938,2	6,81	70,1	5,34	882,8
R-50-40-3	10,31	28.975,3	1,67	81,6	6,17	2.411,0
R-50-40-5	18,60	48.876,6	14,18	140,9	18,41	4.303,1
R-50-100-3	97,29	103.885,7	8,84	43,8	38,47	6.289,1
R-50-100-5	241,39	237.953,3	61,71	216,8	122,49	11.451,2

Table 1: Computational results for the basic case

Instances	mcf branch&bound		branch&cut		lp branch&bound	
	comp. time	#nodes	comp. time	#nodes	comp. time	#nodes
S-10-40-3	0,01	55,6	0,12	0,7	0,08	25,3
S-10-40-5	0,01	51,3	0,21	2,7	0,12	34,4
S-10-100-3	0,03	102,0	0,18	0,9	0,14	17,6
S-10-100-5	0,07	223,7	0,82	2,9	0,39	58,5
S-20-40-3	0,05	233,6	0,29	0,8	0,20	56,4
S-20-40-5	0,10	453,8	1,68	17,4	0,91	255,8
S-20-100-3	0,13	267,1	0,86	2,5	0,78	136,6
S-20-100-5	0,35	672,4	3,47	9,3	2,08	268,8
S-50-40-3	0,24	622,5	1,18	5,7	1,07	196,1
S-50-40-5	0,31	858,5	9,36	71,9	5,47	790,7
S-50-100-3	1,21	1.185,0	2,86	5,8	3,58	290,4
S-50-100-5	2,85	3.002,7	20,08	63,7	15,00	807,3
R-10-40-3	0,01	67,9	0,15	7,5	0,08	24,0
R-10-40-5	0,04	248,5	0,80	20,4	0,30	125,1
R-10-100-3	0,02	40,7	0,20	0,2	0,14	15,1
R-10-100-5	0,20	546,7	2,46	27,0	1,04	213,2
R-20-40-3	0,12	484,5	0,60	21,6	0,32	132,7
R-20-40-5	0,24	1.062,4	2,91	62,2	1,50	505,7
R-20-100-3	0,26	453,0	1,81	26,5	1,27	253,6
R-20-100-5	5,50	9.671,1	11,95	105,2	8,44	1.226,5
R-50-40-3	0,19	526,5	2,38	25,8	1,15	214,0
R-50-40-5	0,56	1.552,2	19,18	273,7	7,32	2.099,7
R-50-100-3	2,12	2.046,3	7,66	34,1	3,69	287,5
R-50-100-5	15,55	15.900,1	59,75	228,9	27,79	1.731,3

Table 2: Computational results for variant 1 (market share constraints)

Instances	mcf branch&bound		branch&cut		lp branch&bound	
	comp. time	#nodes	comp. time	#nodes	comp. time	#nodes
S-10-40-3	0,02	118,7	0,14	0,4	0,06	27,9
S-10-40-5	0,02	166,8	0,49	23,2	0,11	50,6
S-10-100-3	0,02	69,8	0,29	6,8	0,12	17,5
S-10-100-5	0,16	513,3	2,48	66,5	0,40	75,3
S-20-40-3	0,07	389,9	0,45	5,2	0,17	71,3
S-20-40-5	0,43	1.997,3	4,42	148,7	0,61	219,3
S-20-100-3	0,32	749,9	2,08	43,4	0,63	135,7
S-20-100-5	0,74	1.545,9	11,47	164,2	1,26	185,8
S-50-40-3	5,57	16.724,3	8,90	261,9	1,98	739,7
S-50-40-5	36,90	94.865,4	147,56	2.500,0	9,88	2.491,3
S-50-100-3	22,39	26.625,3	28,42	523,7	5,65	832,3
S-50-100-5	171,14	172.466,7	271,80	3.006,0	32,31	3.365,0
R-10-40-3	0,02	55,5	0,10	0,1	0,06	21,6
R-10-40-5	0,05	428,5	0,67	18,5	0,17	82,9
R-10-100-3	0,01	45,6	0,17	0,2	0,10	9,3
R-10-100-5	0,24	1.068,9	2,49	42,6	0,50	91,5
R-20-40-3	0,13	793,8	0,60	15,9	0,24	117,0
R-20-40-5	0,47	3.167,6	3,55	89,8	0,91	343,9
R-20-100-3	0,53	1.369,0	1,85	17,6	1,00	241,4
R-20-100-5	3,11	10.389,8	25,09	434,8	4,46	801,6
R-50-40-3	15,91	59.066,0	35,16	1.195,8	10,51	4.615,8
R-50-40-5	39,33	171.566,6	169,82	1.511,1	25,54	6.714,3
R-50-100-3	130,42	206.668,9	274,82	6.035,4	79,93	14.059,4
R-50-100-5	446,85	798.002,9	2.036,07	17.577,4	398,31	45.945,3

Table 3: Computational results for variant 2 (more for less)

Instances	mcf branch&bound		branch&cut		lp branch&bound	
	comp. time	#nodes	comp. time	#nodes	comp. time	#nodes
S-10-40-3	0,26	1.786,3	0,17	3,4	0,29	372,0
S-10-40-5	0,32	2.135,7	0,27	8,1	0,41	421,1
S-10-100-3	0,73	2.017,0	0,35	8,0	1,23	603,4
S-10-100-5	1,43	4.115,9	1,17	16,5	2,65	1.007,4
S-20-40-3	6,70	30.788,1	0,34	2,4	3,00	2.763,0
S-20-40-5	15,48	66.377,0	1,07	27,3	2,39	1.345,2
S-20-100-3	23,83	44.780,9	1,63	17,3	16,76	5.083,6
S-20-100-5	20,02	36.588,2	2,73	12,8	7,15	1.597,2
R-10-40-3	0,03	229,6	0,09	2,1	0,06	25,6
R-10-40-5	0,43	2.697,7	0,57	35,9	0,26	181,9
R-10-100-3	0,10	302,7	0,11	0,0	0,13	26,6
R-10-100-5	0,71	2.040,6	1,33	28,5	0,74	195,9
R-20-40-3	2,06	9.812,6	0,30	8,5	0,37	252,7
R-20-40-5	5,73	25.467,2	1,63	61,2	1,08	578,6
R-20-100-3	7,07	13.881,7	0,91	9,5	2,38	730,9
R-20-100-5	26,47	48.444,9	6,49	66,6	4,19	920,6

Table 4: Computational results for variant 3 (limited nr. of winning suppliers)