A BRANCH-AND-PRICE ALGORITHM AND NEW TEST PROBLEMS FOR SPECTRUM AUCTIONS

OKTAY GÜNLÜK, LÁSZLÓ LADÁNYI, AND SVEN DE VRIES

Abstract. When combinatorial bidding is permitted in Spectrum Auctions, such as the upcoming FCC auction #31, the resulting winner-determination problem can be computationally challenging. We present a branch-and-price algorithm based on a set-packing formulation originally proposed by Dietrich and Forrest (2002) under contract of the FCC. This formulation has a variable for every possible combination of winning bids for each bidder. Our algorithm exploits the structure of the XOR-of-OR-bidding-language used by the FCC. We also present a new methodology to produce realistic test problems based on the round-by-round-results of FCC auction #4. We generate 2639 test problems which involve 99 items and are substantially larger than most of the previously used benchmark problems. Since there are no real-life test problems for combinatorial spectrum auctions with the XOR-of-OR language, we used these test problems to observe the computational behavior of our algorithm. Our algorithm can solve all but one test problems within ten minutes, appears to be very robust and compares favorably to the natural formulation solved using a commercial optimization package.

A version of our code, extended and adapted to accommodate the details of auction #31, under contract from the FCC has since been delivered to be used as the shadow solver (solver to verify that the solution of the primary solver, which is based on CPLEX, is really optimal) for auction #31.

Keywords: Spectrum Auctions, Test Problems, Combinatorial Auctions, Branch-and-Price

1. Introduction and Background

Traditionally the Federal Communications Commission (FCC) has used administrative hearings to distribute licenses for spectrum use, such as radio, television, wireless telephone

Date: August 14, 2003.
Research partially conducted when the third author was visiting IBM T.J. Watson Research Center.

and satellite. In these hearings, sometimes called "beauty contests", all interested parties would present their business plans to a commission which would then try to select the "best" plan and award licenses to those with the best proposal. This process has been very time consuming, sometimes taking years to complete, and at times has been accused of favoritism due to the subjective nature of the selection process.

Due to the apparent shortcomings of this process, the FCC has tried alternative methods to distribute licenses and for a short period of time, lotteries were used instead of administrative hearings. Even though lotteries are a good way to avoid accusations of favoritism, they clearly would not put the licenses into the hands of those who value them most. Furthermore, these lotteries attracted thousands who hoped to become accidental winners. The after-market ameliorated the inefficiencies of the misallocation but at the cost of making the accidental temporary winners rich.

More recently, the FCC turned to auctions so that licenses would be awarded to those who value them most, and prove it by making high bids for obtaining them. This method is also ideologically more desirable for the FCC since it relies on market forces to ensure economically efficient use of spectrum instead of a central planning body.

At the time of this decision (early 90's), auctions involving the sale of a single good were already a well studied subject, see Klemperer (1999) and Wolfstetter (1996, 1999) for recent surveys. Auctions involving more than one good where bidders have multi-good valuations, however, were less explored by the research community. If many goods are for sale at the same time, some items might be complementary (i.e., a pair of items is more valuable than the sum of its constituents' values) and some other items might be substitutes (i.e., a pair of items is less valuable than the sum of the values of the items) and this might vary across bidders and licenses. Therefore it was necessary for the FCC to find a good mechanisms, for the simultaneous sale of many goods, that supports the expression of some of the interaction. After long discussions with scientists, the FCC decided, as Milgrom

(1995) recalls, to implement a simultaneous ascending auction (SAA) proposed by Milgrom, McAfee, and Wilson.

1.1. **Simultaneous Ascending Auctions.** In an SAA multiple goods are offered for sale, and the process involves several rounds of bidding. In each round, bidders can submit bids for each good separately but simultaneously. After a bidding round is closed the highest bid is determined and announced for each good. In the next round bidders can submit new bids that are higher than the highest bid from the previous round. A minimum increment is required to avoid that bidders top the highest bid only by tiny amounts. Furthermore, to discourage bidders from remaining inactive for a period to observe the behavior of others, there is an "activity rule" that determines how many goods the bidder is allowed to bid in the next round based on his current bids. Therefore if a bidder is inactive in one round, his future activity is limited to a reduced level (unless he uses one of his "waivers"). When no new bids are submitted, the auction ends for all items simultaneously.

Starting in 1994, the FCC used a sequence of auctions for an increasing number of licenses and regions for *Personal Communication Services* (PCS) to test and refine the design of the SAA. In Table 1, we present the following characteristics of the first five PCS auctions (note that actions #2 and #6 were not PCS auctions): bandwidth coverage, number of licenses offered, number of geographical regions, number of qualified bidders, number of winning bidders and total revenue for the FCC.

Auction	PCS Coverage	Licenses	Regions	Bidders	Winners	Revenue
#1	Nationwide narrowband	10	1	29	6	\$650 M
#3	Regional narrowband	30	5	28	9	\$395 M
#4	Broadband blocks A/B	99	51	30	18	\$7,736 M
#5	Broadband block C	493	493	255	89	\$9,270 M
#11	Broadband blocks D/E/F	1472	493	153	125	\$2,523 M

Table 1. First five PCS auctions

All licenses offered in auction #1 were nationwide licenses (with differences in the amount of spectrum they covered) and therefore possible aggregations differed merely in the quality of the coverage but not in the geography of the coverage. Additionally, the FCC permitted each bidder to win at most 3 licenses. PCS auction #3 added the element of geographical regions, and therefore possible aggregations differed both in bands and in geographical regions they covered. The geographical regions in this auction were quite large since the country was divided into only five regions.

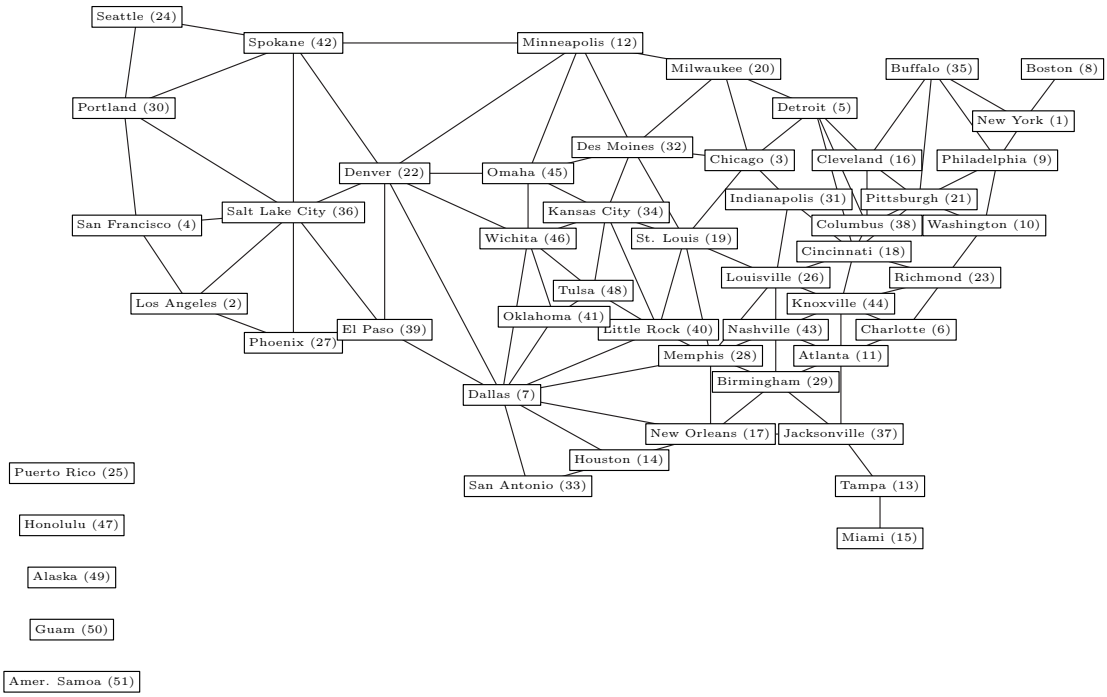

FIGURE 1. Adjacency Graph of MTA's in Auction #4

In the following two auctions the geographic partition of the United States into regions became finer and finer. In auction #4 two licenses in each of the 51 *Major Trading Areas* (MTAs), less 3 licenses awarded separately by pioneer's preference, were for sale. The excluded licenses were for New York, Washington/Baltimore, and Los Angeles/San Diego area. The MTAs were shaped to capture as much geographic interdependence as possible by trying to cover each major city together with its vicinity. See the schematic representation

in Figure 1. In auction #5, one license and in auction #11, three licenses in in each of 493 *Basic Trading Areas* were on sale. Later, there were two more PCS auctions, namely, auction #22 and auction #35 in which blocks of licenses were offered. Details on these auctions can be found on the FCC web site, also see McMillan (1994) and Cramton (1995, 1997, 2002). Additionally, Moreton and Spiller (1998) and Ausubel et al. (1997) both studied the presence of geographic complementarities in auctions #4 and #5. We discuss geographic complementarities in detail later in Section 4.

During the discussion in the early 90's preceding the first FCC spectrum auction, the possibility of permitting bids on packages was discussed. Package bidding allows the bidders to bid on a collection of licenses as a package and therefore facilitates better expression of complementarity and substitutionality (for a recent survey on such "combinatorial auctions" see de Vries and Vohra (2003)). But at that time it was feared that "Package bids are too complex" (Cramton, 1997, Sect. 2.3), see McMillan (1994).

1.2. **Combinatorial Auctions.** The idea underlying combinatorial auctions is based on the observation that bidders might value a set of items differently from the sum of the values of its members. This approach is considered desirable by the FCC since: *"[it] allows bidders to better express the value of any synergies (benefits from combining complementary items) that may exist among licenses and to avoid the risk of winning only part of a desired set,"* see Federal Communications Commission (2001). Once the package, or, combinatorial bids are known, the auctioneer finds a *best allocation* of the licenses to the bidders. This is also known as the winner determination problem and we discuss it further in Section 2. Also see Rothkopf et al. (1998) for further discussion on the difficulty of solving this problem.

1.3. **Simultaneous Ascending Auction with Package Bidding.** In the years 2000/2001 auctions were run in many European countries to sell newly allocated spectrum for Universal Mobile Telecommunications System (UMTS) use. Most of these auctions used elements of the SAA; either following the British approach of pre-packaging spectrum into desirable

packages and then selling these packages to buyers permitted to buy a single package, or the German approach of selling off chunks of spectrum while bidders are allowed to win either two or three chunks. More details on the outcome of these auctions can be found in Binmore and Klemperer (2002), Grimm et al. (2001), Wolfstetter (2001), Jehiel and Moldovanu (2001), Ewerhart and Moldovanu (2001a,b), Klemperer (2002), and de Vries and Vohra (2001).

At around the same time, the FCC was making rules for an auction of similar licenses in the United States, called auction #31. In this auction, one chunk of 20 MHz and another of 10 MHz of spectrum in 6 regions of the United States will be for sale. In total 12 items are going to be sold. Early in 2000 the FCC decided to postpone the decision and reconsider combinatorial (or, package) bidding for auction #31.

During the summer of 2000 the Federal Communications Commission (2000) asked for scientists' comments on permitting package bidding for the upcoming auction #31 to overcome the exposure problem (that is, a bidder bids high singleton bids to obtain a large bundle, but wins only some of them which he values lower than what he bid for them), see Bykowsky et al. (2000). The FCC suggested to permit bids on single items and 9 prespecified subsets (the "global" package, comprising all items; two "national" packages of either all six 10 MHz or all six 20 MHz licenses; six "regional" packages of both, the 10 and the 20 MHz license, in a region) in a *simultaneous ascending auction with package bidding* (SAA-PB). The resulting discussion raised many questions and answered some of them (see http://wireless.fcc.gov/auctions/31/releases.html#comments).

There was some discussion on whether or not it was necessary to restrict the bidding to a few predetermined, agreeable sets. But the opinion to let the bidders themselves determine what is most important to them prevailed.

In addition, it was decided to limit the number of bids of each bidder by allowing them to designate up to 12 packages during the process. In each round bidders would then be permitted to submit bids for each of the designated packages and any single license. For each bidder, bids submitted in the same round would be considered non-exclusive (unless

they were overlapping) whereas bids from different rounds would exclude each other. This way, each bidder can designate his packages so as to overcome the worst exposure problems of his valuations, while not being overburdened by submitting bids for all subsets. In this framework, in round r bidder j submits a clause of the form

$$(S^r_{j,1}, b^r_{j,1}) \text{ OR } (S^r_{j,2}, b^r_{j,2}) \text{ OR } \cdots \text{ OR } (S^r_{j,l(j,r)}, b^r_{j,l(j,r)})$$

where $l(j,r)$ is the number of bids in round r and each pair $(S^r_{j,k}, b^r_{j,k})$ describes a package $S^r_{j,k}$ for which the bidder is revealing willingness to pay $b^r_{j,k}$. Taken together, the bidder signals willingness to take any combination of sets from his clause and to pay for it the corresponding sum. Under the current rules, $S^r_{j,k}$ is either one of the 12 designated packages or a singleton; nevertheless, for our experiments we permit more than 12 packages.

Initially, the FCC ruled that bids submitted in either the current or the immediately preceding round together with older bids that were temporary winners in the immediately preceding round should be considered for the current winner determination problem. Recently (Federal Communications Commission, 2002a) it was ruled that not only the bids from the two previous rounds, but from *all* previous rounds are considered.

For notational simplicity and without loss of generality, we will assume that there are bids in the winner determination problem from rounds 1 through t (of course, some bidders may have empty clauses in some rounds). Thus the bids considered for bidder j are

$$\text{XOR}^t_{r=1} \left[(S^r_{j,1}, b^r_{j,1}) \text{ OR } (S^r_{j,2}, b^r_{j,2}) \text{ OR } \cdots \text{ OR } (S^r_{j,l(j,r)}, b^r_{j,l(j,r)}) \right].$$

This signals that the bidder is only willing to accept bundles from one of the XOR clauses and paying according to what that clause offers. Thus a bidder's bid is an XOR-of-OR-bid.

In the remainder of the paper we concentrate on how to formulate and solve the winner determination problem and argue that with the right formulation and solution algorithm, it is possible to solve these problems even if a large set of items are offered simultaneously.

2. Efficient Allocation / Winner Determination problem

In this Section, we first present a "natural" formulation for the winner determination problem for XOR-of-OR bids. Next, we describe an alternative formulation which proved to be very effective in our computational experiments.

2.1. **Natural formulation of the winner determination problem.** Let N and M denote the set of bidders and items, and assume that there are u_i identical copies of item i available for each $i \in M$. Using the XOR-of-OR notation, the auctioneer's problem at round t reduces to finding an optimal solution to the integer program (1) where $B_{j,r}$ is the set of bids in bidder j's clause from round r.

In formulation (1) indicator variable $x_{j,r,S}$ takes value 1 if bidder i's bid (S, b) submitted in round r is a winning bid. Indicator variable $z_{j,r}$ takes value 1 if bidder j gets some bid of $B_{j,r}$ fulfilled; this condition is ensured by inequality (1b). Furthermore, inequality (1c) ensures that if bidder j gets multiple bids fulfilled they have been submitted during the same round; thereby the XOR-structure of his bid is respected. Inequality (1a) finally ensures that no item is allocated with higher quantity than available.

$$\max \sum_{j\in N}\sum_{r=1}^{t}\sum_{(S,b)\in B_{j,r}} b\, x_{j,r,S}$$

$$\text{(1a)} \qquad \text{s.t.} \sum_{j\in N}\sum_{r=1}^{t}\sum_{(S,b)\in B_{j,r}:S\ni i} x_{j,r,S} \le u_i \qquad \forall i \in M$$

$$\text{(1b)} \qquad x_{j,r,S} \le z_{j,r} \qquad \forall j \in N,\ (S,b) \in B_{j,r},\ r = 1,\dots,t$$

$$\text{(1c)} \qquad \sum_{r=1}^{t} z_{j,r} \le 1 \qquad \forall j \in N$$

$$\text{(1d)} \qquad x_{j,r,S} \in \{0,1\} \qquad \forall S \subseteq M,\ j \in N,\ r = 1,\dots,t$$

A very similar formulation would be obtained when applying the recent transformation technique from symbolic bidding clauses to "concise" formulations by Boutilier (2002).

2.2. **An alternative formulation of the winner determination problem.** In this section we describe an alternative formulation obtained by specializing the formulation presented in Dietrich and Forrest (2002, 2000) to the XOR-of-OR-bids. This formulation is based on the idea of using more variables to get a tighter LP-relaxation (taken to its extreme, Bikhchandani and Ostroy (2002) present several even larger but integral formulations).

For each bidder j and each round r we define $P_{j,r}$ to denote the set of all possible bid combinations that the bidder could win in that round. In other words, an element of $P_{j,r}$ corresponds to a consistent subset of $B_{j,r}$ in the sense that the selected bids can be satisfied simultaneously without violating the availability of the items.

$$P_{j,r} = \left\{ \left(\bigcup_{(S,b)\in\mathcal{I}} S, \sum_{(S,b)\in\mathcal{I}} b \right) : \mathcal{I} \subseteq B_{j,r}, \sum_{(S,b)\in\mathcal{I}} \chi^S \leq (u_1, \ldots, u_m)^\top \right\},$$

where χ^S is the indicator vector of S with respect to the items. In the rest of the paper, we refer to the elements of $P_{j,r}$ as *bundles* or *packages*.

Let $P_j = \bigcup_{r=1}^{t} P_{j,r}$, denote the collection of all valid bundles for bidder j and note that these bundles conform to his submitted XOR-of-OR-bids and satisfy the resource constraints. In formulation (2) we use indicator variable $y_{j,S}$ to denote if bidder j gets bundle $\langle S, b\rangle$. Inequality (2a) ensures that each item is sold only as many times as available, while constraint (2b) enforces that each bidder gets at most one bundle.

$$\max \sum_{j\in N} \sum_{\langle S,b\rangle\in P_j} b\, y_{j,S}$$

$$\text{(2a)} \qquad \text{s.t.} \sum_{j\in N} \sum_{\langle S,b\rangle\in P_j : S\ni i} y_{j,S} \leq u_i \qquad \forall i \in M$$

$$\text{(2b)} \qquad \sum_{\langle S,b\rangle\in P_j} y_{j,S} \leq 1 \qquad \forall j \in N$$

$$\text{(2c)} \qquad y_{j,S} \in \{0,1\} \qquad \forall S \in P_j,\ j \in N$$

Clearly, sets P_j and therefore formulation (2) is of exponential size and hence it is not practical to explicitly generate all possible variables and input this formulation into a general purpose Integer Program (IP) solver. Using branch-and-price, however, one can implicitly consider all possible variables and solve this optimization problem implicitly. See Barnhart et al. (1998) for the idea of branch-and-price and Anbil et al. (1998) for an airline crew-scheduling application. We discuss the details of our approach in Section 3.

Since in the FCC auction we are interested in have one copy of each item, in the rest of the paper we will use $u_i = 1$ for all $i \in M$, even though this approach easily generalizes to $u_i > 1$.

2.3. **Comparison of the formulations.** One very important aspect of a good integer programming formulation is how well its linear programming relaxation approximates the original problem (see Nemhauser and Wolsey, 1988, for a general discussion on this subject). From a computational point of view, the difference between the formulations presented above can be significant, since the LP-relaxation of formulation (2) is stronger and therefore it is likely to result in smaller enumeration trees for branch-and-bound. We provide some experimental evidence supporting this claim in Section 5.1.

To provide some intuition, we next consider a small combinatorial auction that has a single round and a single bidder. The auction is held for three items $\{A, B, C\}$ and the only bidder submits three bids $\{A, B\}$, $\{A, C\}$, and $\{B, C\}$ together with a bid amount of \$15, \$10 and \$10, respectively. Formulation (1) for this problem is:

$$
\begin{array}{llrcll}
\max & 15x_{\{A,B\}} + 10x_{\{A,C\}} + 10x_{\{B,C\}} & & & & \\
& \text{s.t.} & x_{\{A,B\}} + x_{\{A,C\}} & \leq & 1 & \text{(for item } A\text{)} \\
& & x_{\{A,B\}} + x_{\{B,C\}} & \leq & 1 & \text{(for item } B\text{)} \\
& & x_{\{A,C\}} + x_{\{B,C\}} & \leq & 1 & \text{(for item } C\text{)} \\
& & x_{\{A,B\}}, x_{\{A,C\}}, x_{\{B,C\}} & \in & \{0,1\} &
\end{array}
$$

where there is one variable for each bid. The optimal solution to the LP relaxation of this problem assigns a fractional value of 1/2 to each variable to obtain an optimal (fractional) return of \$17.5.

Formulation (2) uses feasible bundles as variables as follows

$$
\begin{array}{lrcll}
\max & 15y_{\{A,B\}} + 10y_{\{A,C\}} + 10y_{\{B,C\}} & & & \\
\text{s.t.} & y_{\{A,B\}} + y_{\{A,C\}} & \leq & 1 & \text{(for item } A\text{)} \\
 & y_{\{A,B\}} + y_{\{B,C\}} & \leq & 1 & \text{(for item } B\text{)} \\
 & y_{\{A,C\}} + y_{\{B,C\}} & \leq & 1 & \text{(for item } C\text{)} \\
 & y_{\{A,B\}} + y_{\{A,C\}} + y_{\{B,C\}} & \leq & 1 & \text{(bundle constraint)} \\
 & y_{\{A,B\}}, y_{\{A,C\}}, y_{\{B,C\}} & \in & \{0,1\} &
\end{array}
$$

where the last inequality is due to the fact that a bidder can at most be awarded one feasible bundle. The optimal solution to the LP relaxation of this formulation assigns a value of 1 to the first bundle only to obtain an optimal return of \$15.

As this example demonstrates, the LP relaxation of formulation (1) is strictly dominated by the LP relaxation of formulation (2) in some cases. The converse, however, is not true since every (fractional) solution to the LP relaxation of formulation (2) is also a solution to the LP relaxation of formulation (1). This is not very surprising since the second formulation contains more information about the problem structure than the first one. The main difficulty in using the second formulation is handling a very large set of variables. In the next section we discuss how this problem can be handled exactly and still effectively. The following lemma makes the dominance precise.

Lemma 1. *If y is a feasible solution for the LP-relaxation of* (2) *then there exists a feasible solution* (x, z) *for the LP-relaxation of* (1) *with the same objective value. However, there*

may be feasible solutions (x, z) *of the LP-relaxation of* (1) *for which there is no solution for the LP-relaxation of* (2) *with the same objective value.*

Proof. The second part follows immediately from the previous example. For the first part consider a feasible solution y to the LP-relaxation of (2). For each nonzero $y_{j,T}$ consider a *decomposition* of the corresponding bundle into bids, i.e., a set of non-intersecting bids such that the union of the items in the bids is the set of items in the bundle and the sum of the costs of the bids is the cost of the bundle. There may be multiple decompositions of a bundle, in such cases we pick one arbitrarily. Denote $Y_{j,T}$ the set of bids in the decomposition of the bundle corresponding to $y_{j,T}$. Now define

$$\begin{aligned} x_{j,r,S} &= \sum_{Y_{j,T} \ni S} y_{j,T} \quad \forall j \in N, \quad r = 1, \ldots, t, \quad \forall S \in B_{j,r} \\ z_{j,r} &= \max_{S \in B_{j,r}} x_{j,r,S} \quad \forall j \in N, \quad r = 1, \ldots, t \end{aligned}$$

With this definition for a fixed item i we prove that x satisfies (1a):

$$\begin{aligned} \sum_{j \in N} \sum_{r=1}^{t} \sum_{S \in B_{j,r}: S \ni i} x_{j,r,S} &= \sum_{j \in N} \sum_{r=1}^{t} \sum_{S \in B_{j,r}: S \ni i} \sum_{Y_{j,T} \ni S} y_{j,T} = \sum_{j \in N} \sum_{Y_{j,T}: T \ni i} y_{j,T} \\ &= \sum_{j \in N} \sum_{(T,b) \in P_j: T \ni i} y_{j,T} \leq u_i. \end{aligned}$$

The definition of z ensures that (1b) holds and it is easy to verify that (x, z) fulfills (1c):

$$\sum_{r=1}^{t} z_{j,r} = \sum_{r=1}^{t} \max_{S \in B_{j,r}} x_{j,r,S} = \sum_{r=1}^{t} \max_{S \in B_{j,r}} \sum_{Y_{j,T} \ni S} y_{j,T} \leq \sum_{r=1}^{t} \sum_{(T,b) \in P_{j,r}} y_{j,T} = \sum_{(T,b) \in P_j} y_{j,T} \leq 1$$

Finally, we must show that x is between 0 and 1. Non-negativity is obvious and

$$x_{j,r,S} = \sum_{Y_{j,T} \ni S} y_{j,T} = \sum_{(T,b) \in P_j} y_{j,T} \leq 1.$$

□

Although, this comparison shows that the LP-relaxation of (2) can be better than that of (1), we note that commercial IP-solvers might add additional constraints thereby strengthening the LP-relaxation of (1). A successful application of our formulation will depend on how the LP-relaxation of (2) compares to that of (1) after strengthening. This implies that the extreme case of plain XOR-type bid structure is not well suited for our formulation since in this case the formulations are identical. We suspect that in cases near this extreme our approach is likely to be slower in plain branch-and-bound than, say, CPLEX. We discuss this further in Section 5.4.2.

3. Branch-and-Price

Branch-and-price (B&P) is a relatively recent and very powerful tool that combines column (or, variable) generation with enumeration to solve large integer programs. B&P can be considered a generalization of linear programming (LP) based branch-and-bound where one partitions the set of feasible solutions into smaller and smaller subsets (branching) and solves the LP relaxation of the problem (bounding) restricted to each subset. This way one can obtain provably optimal solutions.

The novel idea in B&P is to solve LP relaxations of the problem to optimality without explicitly considering all of the variables. Initially only a small subset of the variables are included in the LP formulation to obtain a restricted formulation that can easily be solved to optimality. Based on the *dual* solution to this restricted formulation, the next step is to solve the so-called *pricing problem* to identify new variables that have improving reduced cost (for a definition of reduced cost, see Nemhauser and Wolsey (1988)). If the solution to the pricing problem produces new variables, the current LP formulation is augmented with these variables creating a new formulation with a potentially better objective function value. The search for more variables is then repeated using the dual solution to the new formulation. This procedure terminates when the pricing problem produces a certificate that there are no improving reduced cost variables left out of the formulation.

After making several branching decisions, it is also possible that the current LP formulation, with all possible columns, becomes infeasible. In this case, the current formulation is not explored any further. In Section 3.3, we describe how to detect infeasibility of the complete formulation without generating all columns.

If the LP is solved to optimality using this column generation framework, the next step is to check whether the optimal (primal) solution is integral or not. If the solution is integral, this means that the IP, restricted by the prior branching decisions, is successfully solved to optimality. In this case, the value of the objective function is compared to the best available integral solution, and if the current solution is better, it is declared to be the best available solution for the problem. Otherwise, if the current solution is dominated by a prior solution, it is discarded. If the solution is fractional, then the partition of the solution space is refined by further branching. Our algorithm stops after it finds the first solution it is able to prove optimal.

Next, we describe several branching strategies, their effect on the pricing problem, and then describe how the branching rules can be implemented in terms of formulation (2).

3.1. **Branching rules.** We considered the following four branching rules to partition the solution space, we will discuss them first in terms of formulation (1) and study them for formulation (2) in Sections 3.2 and 3.3. The first two branching rules divide the solution space into two pieces whereas the last two partition the space into several subsets.

(**B1**) Branch on a bid: This branching rule selects a bid of a bidder and based on this bid divides the solution space into two pieces by forcing this bid to be selected in the final solution in one branch and not be selected in the final solution in the other branch. In other words, variables $y_{j,r,S}$ of formulation (1) are implicitly set to one or zero.

(**B2**) Branch on an item: This branching rule selects a license and a bidder and divides the solution space into two pieces by forcing the license to be awarded to the selected bidder in the final solution in one branch and not to be assigned to that bidder in the other branch.

This rule therefore enforces that either exactly one of the bids of a bidder that contain the selected license is a part of the final solution or none of those bids is accepted. In terms of formulation (1), this rule chooses a bidder $j \in N$ and a license $i \in M$ and enforces $\sum_{r=1}^{t} \sum_{(S,b) \in B_{j,r}: S \ni i} y_{j,r,S}$ to be one in one branch and zero in the other. This branching rule was introduced in Ryan and Foster (1981) in a transportation scheduling context.

(**B3**) Multi-way branch on an item: This branching rule selects a license and divides the solution space into several pieces (up to $|N| + 1$) by forcing the license to be awarded to no one in the first branch and to be awarded to a different bidder in remaining branches. In terms of formulation (1), this rule chooses a license $i \in M$ and for each bidder $j \in N$ interested in the license (i.e., $i \in \cup_{r=1}^{t} \cup_{(S,b) \in B_{j,r}} S$) creates a new branch by enforcing $\sum_{r=1}^{t} \sum_{(S,b) \in B_{j,r}: S \ni i} y_{j,r,S} = 1$. An additional branch is created for the case when the license is not awarded to any bidder, i.e., $\sum_{j \in N} \sum_{r=1}^{t} \sum_{(S,b) \in B_{j,r}: S \ni i} y_{j,r,S} = 0$. This branching rule was first proposed in Parker and Ryan (1994) for bandwidth packing problems.

(**B4**) Restricted multi-way branch on an item: This branching rule selects a license and a subset of the bidders interested in that license. New branches are created by assigning the license to one of the chosen bidders. In addition, one more branch is created for the case when the license is not awarded to any one of the chosen bidders. In terms of formulation (1), this rule chooses a license $i \in M$ and a subset $N' \subset N$ and creates a branch for each $j \in N'$ by forcing $\sum_{r=1}^{t} \sum_{(S,b) \in B_{j,r}: S \ni i} y_{j,r,S} = 1$. An additional branch that enforces $\sum_{j \in N'} \sum_{r=1}^{t} \sum_{(S,b) \in B_{j,r}: S \ni i} y_{j,r,S} = 0$ is also created. In our implementation, we chose up to five bidders for this branching rule, i.e. $|N'| \leq 5$.

We note that all four branching rules described above result in a finite enumeration tree. Furthermore, if any one of these rules fail to partition the solution space any further, it means that the solution to the current LP formulation is indeed integral. The only possible exception might happen when one of **B2, B3, B4** is used and if the solution is integral in terms of license assignment to the bidders but not integral in terms of winning bids of bidders. This might only happen if a bidder has several collections of non-conflicting bids

that exactly cover the same set of licenses and have the same total bid amount. Even in this case, using a simplex-based LP-optimization algorithm guarantees that the solution at hand would be an extreme point solution and therefore would be integral in terms of winning bids.

3.2. **The pricing problem.** As noted in Barnhart et al. (1998), branch-and-price can be a viable algorithm only if in every node of the search tree all columns generated in that node satisfy the already made branching decisions. In other words it must be possible to incorporate the branching decisions into the pricing problem. This usually means that the branching decisions (a partition of the feasible solution space) are made in the variable space of the original (natural) formulation and interpreted in the column generation formulation. Our branching strategies, as described above, conform to this idea.

As described earlier in the present section, the goal of the pricing problem is to find an improving variable that is to find a bundle with positive reduced cost. Assume for a moment that no branching has taken place yet. Let π denote the optimal dual vector corresponding to the item restrictions (2a) and ν be the optimal dual vector corresponding to the bidder constraints (2b). The reduced cost of a bundle consisting of a single bid $(S, b) \in B_{j,r}$ is $b - \sum_{i \in S} \pi_i - \nu_j$; while the reduced cost of a general bundle $\mathcal{I} \subseteq B_{j,r}$ is $\sum_{(S,b) \in \mathcal{I}} b - \sum_{(S,b) \in \mathcal{I}} \sum_{i \in S} \pi_i - \nu_j$. Therefore to find the highest reduced cost bundle for a particular bidder j we first need to solve the following vertex packing problem separately for each round r the bidder has bids in:

$$\max \sum_{(S,b) \in B_{j,r}} (b - \sum_{i \in S} \pi_i) y_{j,r,S}$$

$$\text{s.t.} \sum_{(S,b) \in B_{r,s} : S \ni i} y_{j,r,S} \leq 1 \ \forall i \in M \tag{3a}$$

$$y_{j,r,S} = 0, 1 \ \forall S, \ \text{s.t.} \ (S, b) \in B_{j,r} \tag{3b}$$

As we are looking for a bundle $\mathcal{I}$ of positive reduced cost $(\sum_{(S,b) \in \mathcal{I}} b - \sum_{(S,b) \in \mathcal{I}} \sum_{i \in S} \pi_i - \nu_j)$ we have to find a valid point to (3) with value greater than ν_j. The optimal solution to the

vertex packing problem (3) provides the set of bids we need to combine into a bundle to get the bundle with the *highest* reduced cost when we are restricted to choose bids from round r. If the value is greater than ν_j then we have found a column to add to the LP, otherwise, this proves that for this bidder and this round, there is no column to add. Because of the XOR-relation of the bids placed in separate rounds we can find the highest reduced cost bundle by selecting the best of the optimal solutions to the vertex packing problems for different rounds from 1 to t. If any of those bundles has objective function strictly greater than ν_j we have found a column to add. In our implementation, we solve the pricing problem for every bidder and for every round and add at most one new column per bidder. Although the vertex packing problem is $\mathcal{NP}$-complete in general, the pricing problem described above can be solved easily since by the FCC-rules the bidders are not allowed to place too many bids in any individual round and therefore the pricing problem instances have for all rounds comparable size.

Instead of asking the bidders to submit their bids, one can as well send them the dual information and ask them to solve the pricing problem independently on their own, thereby implementing a decentralized auction as observed by de Vries and Vohra (2003).

Now consider the case when branching decisions have been already made.

(**B1**) Branch on a bid: If the bid is forced into the solution then in the vertex packing subproblem where this bid appears we fix the corresponding variable to 1 and in every subproblem we fix every variable to 0 when the corresponding bid intersects the selected bid. For the other branch (when the bid is forced out of the solution) all we need to do is to fix the corresponding variable to 0. Thus in both branches we have just reduced the size of the vertex packing subproblems.

(**B2**) Branch on an item, (**B3**) Multi-way branch on an item, and

(**B4**) Restricted multi-way branch on an item: In each of these cases the individual branches either assign one particular license to a bidder, or disassociate the license from a set of bidders (maybe from everyone). In the first case we need to fix the variables corresponding to those

bids made by *other* bidders that contain the selected license to 0; and for the selected bidder we need to make the constraint corresponding to the selected license an equality constraint. In the second case we just need to fix the variables corresponding to those bids made by excluded bidders that contain the selected license to 0. Thus in such branches we just reduced the size of the vertex packing subproblems and possibly changed an inequality to equality.

Obviously, any combination of branching decisions will only reduce the sizes of the subproblems to be solved and may introduce equalities. Although the pricing problem for a bidder's bids during any round is again an integer program, it is orders of magnitude smaller than the winner determination problem we started with. In our computational experience the subproblems were all trivially solvable.

3.3. **Enforcing branching decisions.** The branching rules described above are all given in terms of formulation (1). However, they can easily be enforced in formulation (2) as well. We have already discussed how to incorporate them into the pricing problem, so whenever new bundles are generated they will all conform to prior branching decisions. To enforce a new branching decision on the current formulation we do the following two steps. First, we remove columns that do not conform to the decision. Second, if the decision involves assigning something (a bid or license) to a bidder then we turn the constraint corresponding to the bidder into an equality constraint.

Enforcing the branching decisions in the pricing problem and removing non-conforming columns from the current problem ensures that in the final solution bidders will not win items they cannot have. Turning the bidder constraints into equality ensures that if a bidder is assigned anything by the branching decisions then he will win a bundle and therefore he will win everything that is assigned to him. Indeed, since the columns conform to the decisions, each column will contain everything that is assigned to him, and we must choose a column.

We also note that enforcing branching decisions might cause the current LP formulation to become infeasible even when these branching decision do not make the complete formulation infeasible. In this case we first need to recover feasibility by augmenting the current formulation. Fortunately, this task is easy, and actually can be solved the same way as the pricing problem is solved. Notice that if the current LP formulation is infeasible then there exists a dual ray proving that infeasibility. To destroy the proof of infeasibility we have to find a column whose inner product with the dual ray is negative. Therefore solving the pricing problem (3) using the dual ray in place of π and disregarding b in the objective function will provide us with such a column. In case this pricing problem is infeasible or it has a non-positive optimum then we can conclude that this set of branching decisions is inconsistent.

3.4. **Extensions of the Bidding Language.** Here we want to point out that a number of extensions proposed to the OR-of-XOR-language can be easily incorporated into the bidder's pricing problem in our formulation using XOR-of-OR-language.

Leyton-Brown (2002, Sect. 2.2.1–2.2.2) considers the application of a fixed offset to an OR-clause. For example, bidder j wants to raise his bids for all bundles from the preceding round by the same fixed amount $\bar{b}$. This can be achieved by first solving (3), then adding $\bar{b}$ to the objective value of the best solution and only then comparing against ν_j.

In another example, if bidder j (or the auctioneer) wants to enforce a budget-constraint or capacity-constraint for bidder j, this constraint needs only to be added to the pricing problem (3) for each round and bidder without changing the larger main problem (2). Therefore it should make the main problem only slightly harder. As well, if the auctioneer wants to enforce capacity-constraints across all bidders, this requires only the addition of a single dummy item to the entire formulation, which causes only small growth of the model.

Another possible modification of the bidding language is to place different restrictions on the set of packages a bidder is permitted to use. The FCC is resolved to permit arbitrary

sets of packages as long as they contain no more than 12 packages. On one hand, the less restricted the set of packages is the better a bidder could express his intended bids. On the other hand, fewer restrictions provide more opportunity to abuse the system by creating opportunities for bidders to park their activity (trying to bid on uninteresting goods in an attempt to hold back bidding on the truly desired goods) and to signal information thru their bids to each other, and to make the overall winner determination problem unduly difficult. Furthermore, imposing fewer restrictions does not force bidders to bid succinctly, thereby again increasing the computational complexity of the overall problem.

Instead of using *size* restrictions, one could impose restrictions on the *structure* of $B_{j,r}$, such that (3) could be solved in polynomial time. For example, if the bids in $B_{j,r}$ form a collection of *non-crossing* sets (that is, for every pair of bids in $B_{j,r}$ either they are disjoint or one contains the other) then (3) is polynomially solvable. This structure is a special case of the bids forming an *interval graph*, and for such structure the vertex packing problem is polynomially solvable (see Nemhauser and Wolsey, 1988). If bids are from a collection of non-crossing sets, this property is maintained after applying multiple branches via **B1**; because forcing a bid out, means only to exclude it from the collection while forcing it in means to remove all overlapping sets. In both cases the system stays non-crossing.

Notice, that even if we consider only auction problems where the bidder's pricing problem (3) is polynomially solvable, this does not imply that the overall problem becomes polynomially solvable. To understand this, consider auctions with increasing number of bidders, each of which is interested only in a single set. Clearly, every bidder's pricing problem is trivially solvable, while the overall problem is no simpler than any other set-packing problem and is therefore $\mathcal{NP}$-hard.

As the bidder's pricing problem on a non-crossing system is computationally easy, it might be preferably to bidders over using rules under which the pricing problem to them is harder.

4. On Test Problems

To evaluate the performance of the presented algorithm, test problems are necessary. Due to the fact that real-world data from auction #31 are not available before that auction closes, one has to consider artificial problem-sets; furthermore, it would be useful to have problems that are (partially) parameterized and larger than those expected for auction #31. Several different test set generators are available for general combinatorial auctions. These generators usually get their names from the random distribution they use, e.g., *random, weighted random, uniform,* and *decay* are described by Sandholm (2002), *binomial* and *exponential* by Fujishima et al. (1999), and *quadratic* by de Vries and Vohra (2003). For an in-depth discussion see Leyton-Brown et al. (2000a).

However, most of these distributions are not very auction specific; instead, they generate set-packing instances with a slight auction motivation. For instance, most do not make differences between bids and bidding behavior of different bidders, but generate just an anonymous list of bids. Furthermore, Sandholm et al. (2000) point out that on some of these distributions CPLEX's (ILOG, 1997) pre-solver simplifies up to 95% of the instances to a point where already the first LP-solution is integral. This indicates that some of these generation schemes produce very easy set packing problems. According to Leyton-Brown et al. (2002, Fig. 3) it seems that the constant problems (usually called uniform) might be the most difficult. However, none of these distributions is based on a realistic model, uses the XOR-of-OR-language or provides any means to generate multi-round bids.

Leyton-Brown et al. (2000a) were probably the first to try to generate valuations based on different models, motivated by various applications. In particular, in the distribution *proximity in space* they generate certain planar random graphs, where items are nodes and bidders have raw values for single nodes and derive added value from obtaining adjacent nodes. For spectrum auctions, this is a very sensible assumption on the structure of synergies,

as Moreton and Spiller (1998) and Ausubel et al. (1997) studied the outcome of FCC auctions #4 and #5. Both report that they found evidence for the presence of local synergies.

Since none of the above generators create multi-round bids nor use the FCC's XOR-of-OR language we set out to generate such problems from a past auction.

In an effort to obtain as realistic as possible test problems, we decided to derive problem instances from the outcome of auction #4, see Federal Communications Commission (2002b). As mentioned in the introduction, this auction was held for individual items corresponding to a license in an MTA. From this source we generated new combinatorial OR-bids from each bidder's bids from a given round, with synergies derived from adjacency in the underlying graph in Figure 1, that models the adjacency of Major Trading Areas (MTA's) (for a map see `http://wireless.fcc.gov/auctions/data/maps/mta.pdf`).

Algorithm 1 describes the algorithm used to generate the bids submitted by each bidder. This algorithm specifies only the set of (location,band) pairs for each bid, the value of the bid will be generated later. Once the bids $\mathcal{C} = \{C_1, \ldots, C_q\}$ are generated for a bidder/round, for each bid C we set the corresponding bid-amount b_C to

$$b_C = \left(\sum_{u \in C} b_u\right) \cdot \left(1 + \frac{\epsilon}{5} \cdot \begin{cases} |C| - 1 & \text{if } |C| \leq 5 \\ 4 + 0.2(|C| - 5) & \text{if } 6 \leq |C| \leq 10 \\ 5 & \text{if } 11 \leq |C| \end{cases}\right). \tag{4}$$

So the bid for a component equals the sum of the bid values for the items in the component times a *synergy term* (the second factor in (4)) that depends only on the number of items (and not on their values) in this component. We choose this function for four reasons: first, it should be nondecreasing, as larger bundles should not incur smaller synergies; secondly it has to be concave, for otherwise large sets are overly favored and the problem boils down to giving one bidder his largest set; thirdly, for the same reason, it should become constant for

ALGORITHM 1. Generation of Bids from FCC-Auction #4

```
for all bidder do
  for all rounds do
    Let U be the set of locations covered by the (location,frequency) pairs submitted
    by the bidder in this round
    Let G be the subgraph of the MTA adjacency graph (see Figure 1) induced by U.
    for all connected components of G do
      Let D_i be the set of pairs with location in this component.
      if |D_i| = 1 then
        submit the D_i (the pair as a single item bid)
      else if 1 < |D_i| ≤ 8 then
        submit D_i and the (|D_i| − 1) sized subsets of D_i
      else
        submit D_i
        if |D_i| < 15 then
          maxnum= |D_i|
        else if 15 ≤ |D_i| < 20 then
          maxnum= 3|D_i|/2
        else if 20 ≤ |D_i| then
          maxnum= 2|D_i|
        end if
        repeat
          Create a bid by taking a random walk (of length generated randomly uni-
          formly between max(3, |D_i|/5) and max(5, (3 ∗ |D_i|)/5)) in the component.
          If this bid is not yet submitted, do so.
        until maxnum different bids generated (or 20|D_i| attempts were made)
      end if
    end for
  end for
end for
```

$|C|$ beyond some threshold; fourth it should be simple. We picked this one and the resulting problems looked reasonably hard.

For our first class of test problems, called *no-singleton,* we let the bidder submit the bid

$$(C_1, b_{C_1}) \text{ OR } (C_2, b_{C_2}) \text{ OR } \ldots \text{ OR } (C_q, b_{C_q}),$$

while for the second class of problems, called *with-singleton* he can augment his bid with singletons to obtain

$$(\mathrm{OR}_{C\in\mathcal{C}}\,(C, b_C)) \quad \mathrm{OR} \quad (u_1, b_{u_1})\ \mathrm{OR}\ (u_2, b_{u_2})\ \mathrm{OR}\ \ldots\ \mathrm{OR}\ (u_p, b_{u_p}),$$

where (u_i, b_{u_i}) are the items the bidder was bidding on in a given round. Finally, to obtain a full test-instance for the branch-and-price code, we generated these bids for all bidders from a set of consecutive rounds (we tried $1, 10, 20, 40$ consecutive rounds) of the 111 rounds of the FCC-auction and we chose different values for ϵ (0.0, 0.02, 0.05, 0.10, 0.25, 0.50, 1.00).

5. Computational Results

5.1. **Test environment and software tools.** We ran all experiments on a Solaris 8 dual processor Sun Blade 2000 with 8.0 GB of memory and 2 Sun UltraSPARC-III+ CPUs running at 900 MHz, each with 8MB cache. All runs were performed on a single CPU.

We based our branch-and-price implementation on COIN/BCP (see Ralphs and Ladányi, 2001), which is an open source, full-featured framework for solving large scale IPs. The implementation of the framework is problem independent and user hooks are used to implement the problem specific parts of the algorithm. We were therefore able to focus on the problem specific details only. COIN/BCP is actively developed under the Common Optimization Interface for Operations Research (COIN-OR) project (see Lougee-Heimer et al., 2001).

To solve the LP relaxations in the bounding phase of the branch-and-price algorithm and to solve the IP problem arising in column generation we have used the CPLEX 8.0 callable library. We note that most of the IP pricing subproblems were solved to integer optimality by integer preprocessing or by just solving their LP relaxation. For the rest, the enumeration tree consisted of only a few nodes.

For the evaluation of the natural formulation we used CPLEX 8.0 with preprocessing and cut generation enabled. In our runs, we set the relative tolerance to 0 and the absolute tolerance to .99 as termination criteria. Setting the relative tolerance was necessary to ensure

that CPLEX finds the *optimal* solution; at its default value of 1×10^{-6} CPLEX would deem solutions optimal if no improvement is possible in the leading 6 digits even though the remaining digits can be significant. In our generation procedure the bid values are integral, therefore we set the second parameter so that CPLEX would stop as soon as it knew that there was no solution at least 0.99 better than the currently best integral solution.

In the column generation formulation we needed to set several tolerances. New columns were added to the formulation if their reduced cost was below CPLEX's zero-tolerance for reduced cost. In the master problem the absolute gap was set to .99 (just like for the natural formulation), while in the subproblems the relative gap was set to 0 and the absolute gap was set to CPLEX's zero-tolerance for reduced cost.

Both codes returned the first solution found with the optimal solution value.

5.2. **Test Problems.** Using the generator described in Section 4 we generated a total of 5278 problems; half of them with singleton bids and the other half without singleton bids while varying the value of ϵ $(0.0, 0.02, 0.05, 0.1, 0.25, 0.5, 1.0)$ and the number of consecutive rounds r considered $(1, 10, 20, 40)$. We note that we did the algorithm-development and data-generation independently and did not fine-tune the algorithm for the test-data. The problems without singleton bids turned out to be considerably easier for all branching rules and therefore we decided to report on the problems with singletons only. We report some characteristics of these problems in Table 2 where each cell contains the minimum/average/maximum value. As the actual numbers are too large, we just present bounds on the number of bundles by determining for each round the bidder bidding for the least and most licenses. Note, that even though the number of bundles is rather huge, our algorithm implicitly considers all of them with the help of the previously described pricing-oracle.

In the rest of this section we first compare how the various branching rules perform in the branch-and-price approach and then examine how the column generation formulation fares against the natural formulation.

	consecutive rounds			
	1	10	20	40
bidders	17/ 23/ 30	19/ 24/ 30	22/ 24/ 30	22/ 25/ 30
bids	183/ 274/ 346	2155/2785/3262	4947/5625/6441	10455/11266/12116
bundles	2^{15}/ −/ 2^{41}	2^{15}/ −/ 2^{41}	2^{15}/ −/ 2^{41}	2^{15}/ −/ 2^{41}

TABLE 2. Statistics for different numbers of consecutive rounds

5.3. **Comparing the branching rules in branch-and-price.** In this section we compare the branching rules presented in Section 3.2 and show that in our code branching rule **B2** performs better than the others. Readers primarily interested in how the column generation formulation with this branching rule compares to the natural formulation should skip to the next section.

In our experiments, we set a time limit of 10 minutes for every run. This was sufficient to finish almost all of the the problems using branching rule **B2**. Branching rule **B3** (multi-way branch on a license) proved to be very inefficient due to the size of the search trees it generates, thus we do not report on it at all. For all branching rules, we used best bound search combined with diving to process the nodes.

In Table 3, we report on the number of problems solved in the root node without any branching, together with the total number of problems unsolved within the time limit for each branching rule. Note that whether or not a problem is solved in the root node does not depend on the branching rule. In each cell of the table, the top line contains the number of problems solved in the root node and the bottom line the number of problems unsolved within 10 minutes for **B1**, **B2** and **B4**, respectively. Cumulative sums across different values of ϵ and different numbers of consecutive rounds are provided as well as an overall sum.

In Table 4 the average running time (in seconds) is reported for the three branching rules. To compute the average running time we had to decide how to include the unfinished runs for this table. We decided to use the cutoff value, that is 600 seconds, which implies that the presented statistics are lower bounds on the actual averages in some cases.

	consecutive rounds (# of problems)				
ϵ	1 (111)	10 (102)	20 (92)	40 (72)	total
0.00	110	102	91	72	375
	0,0, 0	0,0, 0	0,0,0	0,0,0	0,0, 0
0.02	104	93	83	63	343
	0,0, 0	0,0, 0	0,0,0	0,0,0	0,0, 0
0.05	102	88	78	58	326
	0,0, 0	1,0, 0	1,0,0	1,0,0	3,0, 0
0.10	99	82	76	56	313
	0,0, 0	3,0, 0	3,0,0	4,0,1	10,0, 1
0.25	97	75	67	53	292
	0,0, 0	13,0, 0	13,0,0	16,0,0	42,0, 0
0.50	95	63	54	42	254
	0,0, 0	22,0, 0	24,0,1	22,0,1	68,0, 2
1.00	97	66	54	41	258
	0,0, 0	23,0, 0	28,0,1	29,1,6	80,1, 7
total	704	569	503	385	2161
	0,0, 0	62,0, 0	69,0,2	72,1,8	203,1,10

TABLE 3. Number of instances solved in the root node and number of instances not solved within the time limit for **B1**, **B2**, **B4**.

	consecutive rounds				
ϵ	1	10	20	40	total
0.00	0,0,0	5, 5, 5	11,11,11	20,20, 20	8, 8, 8
0.02	0,0,0	6, 5, 5	15,13,14	29,25, 27	11, 9,10
0.05	1,0,0	15, 6, 6	25,15,16	49,23, 28	20,10,11
0.10	1,0,0	36, 7, 8	36,13,16	75,26, 32	33,10,12
0.25	2,0,0	87, 7, 9	106,13,17	147,23, 26	78,10,12
0.50	2,0,0	142,11,15	173,24,30	199,45, 56	119,18,22
1.00	3,0,0	148,12,23	204,31,56	250,69,114	138,24,42
total	1,0,0	63, 8,10	81,17,23	110,33, 43	321,70,93

TABLE 4. Average runtime for **B1**, **B2**, **B4**.

These two tables indicate that independently of the value of ϵ and of the number of rounds considered rule **B2**, that is, branching on an item, performs better than the other branching rules. In the rest of our experiments we only consider rule **B2**.

5.4. **Comparing branch-and-price to the natural formulation.** In this section we compare the column generation formulation with the natural formulation. To solve the natural formulation we use CPLEX with pre-processing and cut generation options enabled

and therefore the actual comparison is between our approach and the natural formulation "strengthened" by CPLEX using preprocessing and cut generation. We believe that this is a reasonable comparison since the natural formulation, being a compact formulation, can be easily strengthened by a commercial solver without using any problem specific information. In our experiments, turning off the cut generation and MIP preprocessing increased the runtime (on the average) by 46%.

5.4.1. *Comparing the formulations on the new test set.* In Table 5, we report the number of problems solved in the root node and the number of problems not solved within the time limit of 10 minutes for both formulations. As seen in the table, both formulations solve most of the 2639 problems in the root node without any enumeration. It is interesting to note that when preprocessing and cut generation is turned off for CPLEX runs, this number decreases drastically. In terms of solving problems within the time limit, column generation approach performs better than the natural formulation and it appears to scale better as the synergies between the items and as the number of consecutive rounds grow.

		ϵ							consecutive rounds				total
		0.00	0.02	0.05	0.10	0.25	0.50	1.00	1	10	20	40	
colgen	in root	375	343	326	313	292	254	258	704	569	503	385	2161
	unsolved	0	0	0	0	0	0	1	0	0	0	1	1
natural	in root	355	348	332	302	269	243	234	769	572	445	297	2083
	unsolved	1	0	0	0	2	6	32	0	1	6	34	41

TABLE 5. Problems solved in the root and unsolved within the time limit

For the remaining comparisons we have solved all problems to optimality with both solvers. In Table 6 we compare the average and maximum tree size and in Table 7 we compare the average running time for the two formulations. Both tables indicate that the natural formulation has scaling problems whereas the column generation approach is quite robust. Even though both formulations solve a comparable number of problems in the root node, both the average and the maximum number of search tree nodes are much higher for the natural formulation.

		ϵ							consecutive rounds				total
		0.00	0.02	0.05	0.10	0.25	0.50	1.00	1	10	20	40	
colgen	avg	1	1	1	2	3	6	10	1	4	4	6	4
	max	5	11	9	77	47	183	219	23	169	193	219	219
natural	avg	43	4	6	8	83	168	621	1	94	114	419	133
	max	8274	639	951	422	6262	5920	37649	4	37649	5920	35530	37649

TABLE 6. Comparing the average and maximum tree size (no time limit).

		ϵ							consecutive rounds				total
		0.00	0.02	0.05	0.10	0.25	0.50	1.00	1	10	20	40	
colgen	avg	8	9	10	10	10	18	25	0	8	17	33	13
	max	103	119	116	347	118	541	843	7	164	296	843	843
natural	avg	8	3	4	6	22	57	210	0	11	34	172	44
	max	857	119	96	123	1344	1131	7807	0	1894	1295	7807	7807

TABLE 7. Comparing the average and maximum runtime (no time limit).

The last table in this section, Table 8, provides a possible explanation as to why the column generation approach is more successful than the natural formulation. In this table, we compare the average and maximum integrality gap at the root node for both formulations and show that the gap is significantly larger for the natural formulation. We note that the LP value of the natural formulation is taken after preprocessing and cut generation. We have computed the integrality gap for every problem as

$$100 \times \frac{\text{LP value at root} - \text{IP optimum value}}{\text{LP value at root}}.$$

		ϵ							consecutive rounds				total
		0.00	0.02	0.05	0.10	0.25	0.50	1.00	1	10	20	40	
colgen	avg	0.00	0.00	0.00	0.01	0.05	0.15	0.30	0.02	0.08	0.10	0.11	0.07
	max	0.00	0.03	0.12	0.34	0.85	2.13	2.47	1.60	2.47	2.47	2.42	2.47
natural	avg	0.02	0.03	0.06	0.11	0.35	0.81	1.49	0.05	0.46	0.56	0.71	0.41
	max	0.26	0.23	0.35	0.74	1.73	3.67	5.29	2.75	5.27	5.29	4.83	5.29

TABLE 8. The average and maximum gaps (no time limit).

5.4.2. *Further experiments with the formulations.* We performed further computational experiments to compare the natural formulation to the column generation approach using existing problem generators developed for combinatorial auctions. In particular, we used the

proximity in space generator (see Leyton-Brown et al. (2000b)) and a *uniform generator* (see Sandholm (2002)) that we describe below. For these test problems the natural formulation performed better than the column generation formulation. We have tried various problem sizes and solved a number of problems created by these generators and the results were consistently in favor of using CPLEX on the natural formulation. We next discuss why we think the column generation approach performs poorly on these problems.

Proximity in Space Generator: This generator appears to generate reasonably hard set packing problems by generating individual bids random walks on a grid. A somewhat simplified description of the algorithm is the following: For each bidder, first a *private valuation* is generated for each item. Then, a base bid is created using a random walk. Bids are then generated by starting a new random walk from each item in this base bid. Out of the generated bids the base bid and a small number of most valuable bids are kept (we used the default value of five in our experiments). Since the random walk is biased towards the items with higher private valuation the generated bids are likely to be pairwise intersecting. Furthermore, keeping more valuable bids results in keeping bids that contain a larger number of items, which further increases the likelihood of producing intersecting bids. In this setting, it is very unlikely that several bids can be combined into columns for the column generation formulation and therefore the comparison between the two formulations becomes a comparison between how fast the enumeration is done. In our experiments, we observed the behavior described above: no columns were generated and CPLEX was able to enumerate branch-and-bound nodes faster than our code.

Uniform Generator: This generator generates bids by selecting a small number of items uniformly (we used the default value of five in our experiments). In this case, the bids can indeed be combined into columns and we expected the column generation formulation to outperform the natural formulation. However, in our computational experiments we have observed that the value of the LP-relaxation of the column generation formulation was only minimally better than that of the natural formulation and CPLEX cut generation was able

to produce very effective cuts to strengthen the natural formulation. Since it starts off with a worse relaxation, it is not surprising that column generation formulation performs poorly for these problems. We do not know why the column generation formulation is just barely stronger and how the cut generation can completely offset and exceed the gain so easily.

Conclusion and Future Research Direction

In this paper we have presented a column generation based algorithm to solve the winner determination problem given in the XOR-of-OR-language and a methodology to generate realistic test problems.

The test suite appears to exercise the solvers reasonably well, the problems are not impossible to solve, yet they are not overly easy either; a fair amount of column/cut generation and/or branching is necessary to solve all problems to optimality. The harder problems are available over the Internet, see Günlük et al. (2003). For the full test suite interested researchers should contact the authors.

Our algorithm manages to solve almost all of these instances within 10 minutes and performs better than the natural formulation solved using CPLEX. Our computational experiments also indicate that column generation approach is not always superior: If no columns can be generated, or if the LP-relaxation of the column generation formulation is not sufficiently stronger than that of the natural formulation, then solving the natural formulation with CPLEX is faster. However, when columns can be generated and the formulation is stronger (which appears to be the case for our generator mimicking the setup of FCC auction #31) our approach should be seriously considered.

Acknowledgments

The authors are thankful to Brenda Dietrich and Jane L. Snowdon for facilitating this work as well as to John J. Forrest for fruitful discussions. We would also like to thank the anonymous referees whose comments helped clarify our presentation.

References

Anbil, Ranga, John J. Forrest, and William R. Pulleyblank. 1998. Column generations and the airline crew pairing problem. *Doc. Math. J. DMV, Extra Volume ICM 1998* **III** 677–686.

Ausubel, Lawrence M., Peter C. Cramton, R. Preston McAfee, and John McMillan. 1997. Synergies in wireless telephony: Evidence from the broadband PCS auction. *Journal of Economics and Management Strategy* **6**(3) 497–527.

Barnhart, Cynthia, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. 1998. Branch and price: Column generation for solving huge integer programs. *Operations Research* **46**(3) 316–329.

Bikhchandani, Sushil and Joseph M. Ostroy. 2002. The package assignment model. *Journal of Economic Theory* **107**(377–406).

Binmore, Ken and Paul Klemperer. 2002. The biggest auction ever: the sale of the British 3G telecom licences. *Economic Journal* **112**(478) C74–C96.

Boutilier, Craig. Solving concisely expressed combinatorial auction problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, Edmonton, Alberta, CA*, pages 359–366. AAAI Press, 2002.

Bykowsky, Mark M., Robert J. Cull, and John O. Ledyard. 2000. Mutually destructive bidding: The FCC auction design problem. *Journal of Regulatory Economics* **17** 205–228.

Cramton, Peter C.. 1995. Money out of thin air: The nationwide narrowband PCS auction. *Journal of Economics and Management Strategy* **4**(2) 267–343.

Cramton, Peter C.. 1997. The FCC spectrum auctions: An early assessment. *Journal of Economics and Management Strategy* **6**(3) 431–495.

Cramton, Peter C. Spectrum auctions. In Martin Cave, Sumit Majumdar, and Ingo Vogelsang, editors, *Handbook of Telecommunications Economics*. Elsevier Science B.V., Amsterdam, The Netherlands, 2002. URL `http://www.cramton.umd.edu/papers2000-2004/01hte-spectrum-auctions.pdf`. forthcoming.

de Vries, Sven and Rakesh V. Vohra. 2001. Auctions and the German UMTS-auction. *Mitteilungen der DMV* (1) 31–38.

de Vries, Sven and Rakesh V. Vohra. 2003. Combinatorial auctions: A survey. *Informs Journal on Computing* **15** 26. to appear.

Dietrich, Brenda and John J. Forrest. A method for determining the set of winning bids in a combinatorial auction. Patent filed, YOR92000047US1, July 2000.

Dietrich, Brenda and John J. Forrest. A column generation approach for combinatorial auctions. In Brenda Dietrich and Rakesh V. Vohra, editors, *Mathematics of the Internet: E-Auction and Markets*, volume 127 of *The IMA Volumes in Mathematics and its Applications*, pages 15–26. Springer-Verlag, New York, NY, 2002.

Ewerhart, Christian and Benny Moldovanu. The German UMTS design: Insights from multi-object auction theory. URL `http://www.sfb504.uni-mannheim.de/~ewerhart/cesifo.pdf`. Manuscript, 2001a.

Ewerhart, Christian and Benny Moldovanu. A stylized model of the German UMTS auction. URL `http://www.sfb504.uni-mannheim.de/~ewerhart/stylized.pdf`. Manuscript, 2001b.

Federal Communications Commission. Auction of licenses in the 747–762 and 777–792 MHz bands scheduled for september 6, 2000: Comment sought on modifying the simultaneous multiple round auction design to allow combinatorial (package) bidding. URL `http://wireless.fcc.gov/auctions/31/releases/da001075.pdf`. DA 00–1075, May 2000.

Federal Communications Commission. On package bidding. URL http://wireless.fcc.gov/auctions/about/auctiondesigns.html. December 2001.

Federal Communications Commission. Auction of licenses in the 747–762 and 777–792 MHz bands scheduled for june 19, 2002; further modification of package bidding procedures and other procedures for auction no. 31. URL http://wireless.fcc.gov/auctions/31/releases/da020659.pdf. DA 02–659, March 2002a.

Federal Communications Commission. Round-by-round results. URL ftp://ftp.fcc.gov/pub/Auctions/PCS/Broadband/MTA/index.html. May 2002b.

Fujishima, Yuzo, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of IJCAI-99, Stockholm, Sweden*, pages 548–553, Stockholm, 1999.

Grimm, Veronika, Frank Riedl, and Elmar Wolfstetter. The third generation (UMTS) spectrum auction in Germany. URL http://papers.ssrn.com/sol3/delivery.cfm/SSRN_ID287846_code011113590.pdf?abstractid=287846. Manuscript, Institute of Economic Theory I, Humboldt University of Berlin, Berlin, Germany, 2001.

Grötschel, Martin, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics*. Springer Verlag, Berlin Heidelberg, Germany, 1. edition, 1988.

Günlük, Oktay, László Ladányi, and Sven de Vries. Hard problems. URL http://www-m9.ma.tum.de/~devries/fcc_supplement/. April 2003.

ILOG. *Using the CPLEX Callable Library*. Inc. CPLEX Division, 930 Tahoe Blvd. # 802-279, Incline Village, NV 89451-9436, USA, 1997. URL http://www.cplex.com.

Jehiel, Philippe and Benny Moldovanu. The European UMTS/IMT-2000 license auction. URL http://www.vwl.uni-mannheim.de/moldovan/papers/umts1.pdf. Discussion paper, Faculty of Economics, University of Mannheim, Mannheim, Germany, 2001.

Klemperer, Paul. July 1999. Auction theory: A guide to the literature. *Journal of Economic Surveys* **13**(3) 227–286.

Klemperer, Paul. 2002. What really matters in auction design. *Journal of Economic Perspectives* **16**(1) 169–190.

Leyton-Brown, Kevin. Response to Prof. Milgrom and Prof. Ausubel's "Comments on the Second Wye River package bidding conference". URL http://wireless.fcc.gov/auctions/conferences/combin2001/papers/FCC-Comments.pdf. January 2002.

Leyton-Brown, Kevin, Eugene Nudelman, and Yoav Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In Pascal Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002, Ithaca, NY, USA*, volume 2470 of *Lecture Notes in Computer Science*, pages 556–572. Springer, 2002. URL http://link.springer.de/link/service/series/0558/bibs/2470/24700556.htm.

Leyton-Brown, Kevin, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings ACM Conference on Electronic Commerce (EC-00), Minneapolis, MN*, pages 66–76, 2000a. URL http://robotics.Stanford.EDU/~kevinlb/CATS.pdf.

Leyton-Brown, Kevin, Yoav Shoham, and Moshe Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, Austin, TX*, pages 56–61, 2000b. URL http://robotics.Stanford.EDU/~kevinlb/CATS.pdf.

Lougee-Heimer, Robin, Francisco Barahona, Brenda Dietrich, J.P. Fasano, John Forrest, Robert Harder, László Ladányi, Tobias Pfender, Theodore Ralphs, Matthew Saltzman, and Katya

Scheinberg. 2001. The COIN-OR initiative: Open-source software accelerates operations research progress. *OR/MS Today* **28** 20–22.

McMillan, John. 1994. Selling spectrum rights. *Journal of Economic Perspectives* **8**(3) 145–162.

Milgrom, Paul R. *Auction Theory for Privatization*, chapter Auctioning the Radio Spectrum. Cambridge University Press, 1995.

Moreton, Patrick S. and Pablo T. Spiller. 1998. What's in the air: Interlicense synergies in the federal communications commission's broadband personal communication service spectrum auction. *Journal of Law and Economics* **XLI** 677–716.

Nemhauser, George L. and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley, New York, 1988.

Parker, Mark and Jennifer Ryan. 1994. A column generation algorithm for bandwidth packing. *Telecommunications Systems* **2** 185–196.

Ralphs, Ted K. and Laszlo Ladányi. *COIN/BCP User's Manual*, January 2001. URL `http://www.coin-or.org/Presentations/bcp-man.pdf`. 70 pages.

Rothkopf, Michel H., Aleksandar Pekec, and Ronald M. Harstad. 1998. Computationally manageable combinatorial auctions. *Management Science* **44**(8) 1131–1147.

Ryan, David M. and B.A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport: Urban passenger vehicle and crew scheduling*, LNCS, pages 269–280. North-Holland, 1981.

Sandholm, Tuomas W.. 2002. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence* **135**(1–2) 1–54.

Sandholm, Tuomas W., Subhash Suri, Andrew Gilpin, and David Levine. CABOB: a fast optimal algorithm for combinatorial auctions. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Seattle, WA, 2000. URL `http://www-2.cs.cmu.edu/~sandholm/cabob.ijcai01.pdf`. 7 pages.

Wolfstetter, Elmar. 1996. Auctions: An introduction. *Journal of Economic Surveys* **10**(4) 367–420.

Wolfstetter, Elmar. *Topics in Microeconomics: Industrial Organiziation, Auctions, and Incentives*. Cambridge University Press, 1999.

Wolfstetter, Elmar. The Swiss UMTS spectrum auction flop: Bad luck or bad design? URL `http://www.wiwi.hu-berlin.de/institute/wt1/papers/2001/swiss_umts_flop.pdf`. Manuscript, 2001.

Oktay Günlük, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, N.Y. 10598, U.S.A.
E-mail address: `oktay@watson.ibm.com`

László Ladányi, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, N.Y. 10598, U.S.A.
E-mail address: `ladanyi@us.ibm.com`

Sven de Vries, Zentrum Mathematik, TU München, D-80290 München, Germany.
E-mail address: `devries@ma.tum.de`