A Fast Approximation Algorithm for Solving the Complete Set Packing Problem

Tri-Dung Nguyen
Centre of Operational Research, Management Sciences and Information Systems
University of Southampton, Southampton, UK, SO173TJ
T.D.Nguyen@soton.ac.uk

We study the complete set packing problem (CSPP) where the family of feasible subsets may include all possible combinations of objects. This setting arises in applications such as combinatorial auctions (for selecting optimal bids) and cooperative game theory (for finding optimal coalition structures). Although the set packing problem has been well-studied in the literature, where exact and approximation algorithms can solve very large instances with up to hundreds of objects and thousands of feasible subsets, these methods are not extendable to the CSPP since the number of feasible subsets is exponentially large. Formulating the CSPP as an MILP and solving it directly, using CPLEX for example, is impossible for problems with more than 20 objects. We propose a new mathematical formulation for the CSPP that directly leads to an efficient algorithm for finding feasible set packings (upper bounds). We also propose a new formulation for finding tighter lower bounds compared to LP relaxation and develop an efficient method for solving the corresponding large-scale MILP. We test the algorithm with the winner determination problem in spectrum auctions, the coalition structure generation problem in coalitional skill games, and a number of other simulated problems that appear in the literature.

Key words: Set Packing Problem; Combinatorial Auctions; Winner Determination Problem; Coalition Structure Generation; Large-scale MILP.

1. Introduction and Literature Review

1.1. The Set Packing Problem (SPP)

The set packing problem and its variants (the set covering and set partitioning problems) are among the most well-studied problems in combinatorial optimisation thanks to their wide ranges of applications, their elegant mathematical formulation, and their special structural properties. In the SPP, there are n objects which can be packed into a number of subgroups among m predefined feasible subsets labelled as $\mathcal{S}_1, \ldots, \mathcal{S}_m$. Each subset $\mathcal{S}_j$ has a payoff value of v_j. The SPP aims to divide these n objects into non-overlapping subgroups such that their total payoff is maximised. The problem has many applications such as for routing and scheduling trains at intersections in railway operations (Zwaneveld et al. [35]), for selecting winning bids in combinatorial auctions (De Vries and Vohra [16]), for surgical operations scheduling (Velásquez and Melo [34]), and for packets scheduling and transmission in communication networks (Emek et al. [18]) among many others. In the context of combinatorial auctions, the winner determination problem (WDP) is essentially a set packing problem. Sandholm [32] develops an algorithm that utilises the graphical representation of the coalition structure search space for solving the WDP. Fujishima et al. [19] develop an exact algorithm, where caching and pruning are used to speed up the search, and a heuristic algorithm for solving the WDP.

The tractability of the SPP depends on the structure of the underlying IP formulation. Specifically, Müller [28] and Rothkopf et al. [31] summarise special cases where the corresponding LP relaxation solutions satisfy the integrality constraints and hence are also solutions of the SPP. These are, however, very restrictive cases and it is generally very difficult to solve the SPP. In fact, Karp [23] shows that the SPP problem is NP-complete while Sandholm [32] shows the inapproximability of the problem for general cases. Many methods, both exact and approximation, have been proposed for solving the SPP. Padberg [29] and Cánovas et al. [7] show different sets of facets of the set packing polyhedron which can be used to strengthen the LP relaxation solutions. Landete et al. [24] present an alternative formulation for the SPP in a higher-dimensional space where a set of facets can be identified.

Methods for solving the SPP often start with solving the corresponding LP relaxation problem. De Vries and Vohra [16] survey different methods such as a constraint generation method for solving the LP relaxation problem and a sub-gradient method for solving the Lagrangian relaxation. The authors also provide interesting insights on how the numerical algorithm is interpreted in the auctioning process. These methods have actually been well-studied in the context of the set covering problem (SCP), a variant of the SPP where the objective is to minimise the total cost of covering all the objects (see Beasley [4] and Beasley and Jörnsten [6] for examples). Caprara et al. [10] survey methods to solve the SCP and compare their numerical performance on test problems that appear in the literature.

There are also many heuristic methods for solving the SPP. In fact, Hoffman and Padberg [22] state that "virtually every heuristic approach for solving general integer programming problems has

been applied to the set-covering, packing and partitioning problems." Delorme et al. [17] develops GRASP, a greedy randomised algorithm for solving the set packing problem. Beasley and Chu [5] develop a genetic algorithm for solving the set covering problem and this method can be adapted to solve the SPP.

1.2. The Complete Set Packing Problem (CSPP)

In this paper, we aim to solve the SPP for cases when $m = 2^n$, i.e. any subset of objects can be grouped together in a packing, or when m is relatively large compared to n. This setting arises in applications such as combinatorial auctions where bidders submit their bids in the form of value functions on the objects selected. This bidding mechanism is favourable to auction designers and to bidders because the information can be communicated in a more compact way. Another application area is in multi-agent systems, e.g. in a sensor network [14], where players are grouped into coalitions to maximise their total utility. As the number of possible subsets can grow exponentially, existing methods (such as [6, 10, 19, 32]) are not applicable due to the large number of binary decision variables involved.

Let $\mathcal{N} = \{1, \ldots, n\}$ be the set of all objects and let $\boldsymbol{x} = (x_1, x_2, \ldots, x_{2^n})$ be a vector of binary variables with x_j indicating whether subset $\mathcal{S}_j$ is selected in the packing. The CSPP can be formulated as an MILP as follows:

$$\begin{aligned} \mathbf{CSPP}(\mathcal{N}, \boldsymbol{v}) := \max_{\boldsymbol{x}} \ & \boldsymbol{v}^t \boldsymbol{x} \\ \text{s.t.} \ & \boldsymbol{A}_{\mathcal{N}} \boldsymbol{x} \leq \boldsymbol{e}, \\ & \boldsymbol{x} \in \{0,1\}^{2^n}, \end{aligned} \tag{1}$$

where $\boldsymbol{A}_{\mathcal{N}} \in \mathbb{R}^{n \times 2^n}$ is a matrix with element a_{ij} in row i and column j indicating whether subset $\mathcal{S}_j$ contains object i, and $\boldsymbol{e} \in \mathbb{R}^n$ is a vector with all elements being equal to one. For convenience in notation, let $\boldsymbol{a}_j = (a_{1j}, \ldots, a_{nj})^t$ be a column vector of binary indicators for each $j \in \{1, .., 2^n\}$. To avoid ambiguity in the ordering of $\boldsymbol{a}_j$, we assign $\boldsymbol{a}_j$ to the binary representation of $(j-1)$. Let us denote $v_j \equiv v(\boldsymbol{a}_j) \equiv v(\mathcal{S}_j)$ as the payoff of subset $\mathcal{S}_j$. For $n \leq 15$, problem $\mathbf{CSPP}(\mathcal{N}, \boldsymbol{v})$ can be solved efficiently by CPLEX through a classical branch and bound technique. However, the size of the MILP problem grows exponentially as the number of objects increases and it is impossible for CPLEX to solve instances with more than 20 objects. We aim to develop an approximation method for solving this MILP.

1.3. The Winner Determination Problem in Combinatorial Auctions

Combinatorial auctions have been used in the procurement of London bus routes (Cantillon and Pesendorfer [8]), radio spectrum (Cramton [12]), and truckload transportation (Caplice and Sheffi [9]), among many others. Combinatorial auctions arise in situations where bidders are interested in buying bundles of objects that inherit some level of synergies among themselves. One of the key problems in combinatorial auction is to find the best feasible combination of bids to maximise the total payoff. This problem is equivalent to a complete set packing problem where objects are

those to be sold and the payoff of each subset is the maximum bid that the bidders offer. In the combinatorial auction literature, solution approaches such as Fujishima et al. [19] and Sandholm [32] often assume that the number of bids are relatively small compared to the number of objects, i.e. a few hundreds of objects and a few thousands of bids at most. However, in many real-life situations such as in spectrum auctions, bidders might be interested in buying any subset of their predefined frequencies. In this case, the bidders may express their interest through a compact value function that involves their objects of interest and their specific synergy parameters (Cramton et al. [13] and De Vries and Vohra [15]). Therefore, the set of feasible bids from all the bidders is an exponential function of the number of objects. We discuss about one such case in spectrum auctions in subsection 3.1.

1.4. The Optimal Coalition Structure Generation Problem in Cooperative Game Theory

Cooperative games with transferable utilities belong to a branch of game theory where groups of players can form coalitions in order to jointly achieve the groups' objectives. Cooperative game theory has many applications in economics and business (e.g. for setting insurance premiums [26], and for setting interchange fees for ATM bank networks [20]), in law and political science (e.g. for computing voting power [25]), and in artificial intelligence (e.g. for coalition structure formation in multi-agent systems [11]), among many others. One of the key problems in coalitional games is to find a coalition structure, i.e. to divide the set of all players into disjoint subsets called coalitions, such that the total payoff of these coalitions is maximised. This problem is equivalent to a CSPP where players are viewed as objects, coalitions are viewed as subgroups, and a coalition structure is equivalent to a packing. Sandholm et al. (1999) present a coalition structure graph to visualise the set of all possible coalition structures. The authors then show interesting results about the guaranteed bound on the best coalition structure within certain parts of the graph. Since then, new exact methods have been introduced to exploit the special search space of the coalition structures. However, these existing methods are only applicable for games with less than 30 players (Rahwan et al. [30].)

1.5. Contributions

In this manuscript, we develop an approximation method for solving the CSPP. Our contributions include the following:

1. We propose a new mathematical formulation for the CSPP that makes use of the subsets suggested by the LP solution (or any heuristic solution and their combination). The new formulation directly suggests an efficient method for generating near-optimal feasible packings.

2. We propose a method to find tighter upper bounds (compared to LP relaxation). This involves a constraint generation framework to solve the corresponding large-scale MILP problem.

3. We demonstrate the algorithm with the winner determination problem that arises in spectrum auctions. We show that the constraint generation problem can be solved in $\mathrm{O}(n^2 \log n)$ and that the LP relaxation problem can be solved in polynomial time. We provide numerical results for instances with up to 200 objects.

4. We also perform numerical tests on the optimal coalition structure generation problem that arises in large weighted coalitional skill games (Bachrach et al. [2]) and a number of other simulated combinatorial auction settings that are accompanied by underlying economical interpretations (Leyton-Brown and Shoham [27].)

The structure for the rest of the paper is as follows. We provide an alternative formulation for the CSPP problem in subsection 2.1. This leads to an efficient method for finding feasible packings in subsection 2.2. The upper bounds are obtained by solving the LP relaxation and a large-scale MIP relaxation. These are done via constraint generation frameworks to be described in subsections 2.3 and 2.4. We demonstrate the algorithm through two applications in combinatorial auctions and cooperative games in subsections 3.1 and 3.2. We provide numerical results in Section 4 and finally conclude in Section 5.

2. Fast Approximation Algorithm for Solving the CSPP

2.1. Alternative Formulation for CSPP

Let $\mathcal{I}$ be the indices of columns that we have a high expectation on where the optimal set of subsets will lie on, and let $\mathcal{J}$ be the indices of the remaining columns. For now, we assume that $\mathcal{I}$ and $\mathcal{J}$ are given. The CSPP can be reformulated as:

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}_\mathcal{I}, \boldsymbol{x}_\mathcal{J}} & \boldsymbol{v}_\mathcal{I}^t \boldsymbol{x}_\mathcal{I} + \boldsymbol{v}_\mathcal{J}^t \boldsymbol{x}_\mathcal{J}, \\ \text{s.t.} & \boldsymbol{A}_\mathcal{I} \boldsymbol{x}_\mathcal{I} + \boldsymbol{A}_\mathcal{J} \boldsymbol{x}_\mathcal{J} \leq \boldsymbol{e}, \\ & \boldsymbol{x}_\mathcal{I} \in \{0,1\}^{|\mathcal{I}|}, \boldsymbol{x}_\mathcal{J} \in \{0,1\}^{|\mathcal{J}|}, \end{array} \tag{2}$$

where the matrix $\boldsymbol{A}$, cost vector $\boldsymbol{v}$ and decision variable $\boldsymbol{x}$ are divided into two subsets according to indices $(\mathcal{I}, \mathcal{J})$, i.e.

$$\left[\begin{array}{c} \boldsymbol{A} \\ \boldsymbol{v} \\ \boldsymbol{x} \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{A}_\mathcal{I} & \boldsymbol{A}_\mathcal{J} \\ \boldsymbol{v}_\mathcal{I} & \boldsymbol{v}_\mathcal{J} \\ \boldsymbol{x}_\mathcal{I} & \boldsymbol{x}_\mathcal{J} \end{array}\right].$$

Problem (2) can be reformulated as a bi-level optimisation problem:

$$\begin{array}{lll} \max\limits_{\boldsymbol{x}_\mathcal{I} \in \{0,1\}^{|\mathcal{I}|}} & \boldsymbol{v}_\mathcal{I}^t \boldsymbol{x}_\mathcal{I} + & \max\limits_{\boldsymbol{x}_\mathcal{J} \in \{0,1\}^{|\mathcal{J}|}} \boldsymbol{v}_\mathcal{J}^t \boldsymbol{x}_\mathcal{J}, \\ & & \boldsymbol{A}_\mathcal{J} \boldsymbol{x}_\mathcal{J} \leq \boldsymbol{e} - \boldsymbol{A}_\mathcal{I} \boldsymbol{x}_\mathcal{I}, \\ & \boldsymbol{A}_\mathcal{I} \boldsymbol{x}_\mathcal{I} \leq \boldsymbol{e}. & \end{array} \tag{3}$$

Let us denote $f_{IP}^*(\mathcal{N}, \boldsymbol{v})$ as the optimal value of $\mathbf{CSPP}(\mathcal{N}, \boldsymbol{v})$. We also use shorthand notation $f_{IP}^* \equiv f_{IP}^*(\mathcal{N}, \boldsymbol{v})$ where there is no confusion for not specifying $(\mathcal{N}, \boldsymbol{v})$. For each choice of $\boldsymbol{x}_\mathcal{I}$, let $\mathcal{N} \backslash \boldsymbol{A}_\mathcal{I} \boldsymbol{x}_\mathcal{I}$ be the set of remaining objects after subsets in $\boldsymbol{A}_\mathcal{I}$ have been selected according to the

indicator vector $\boldsymbol{x}_{\mathcal{I}}$. Let $\boldsymbol{v}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}$ be the corresponding reduced vector of payoffs that subgroups of $\mathcal{N}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}$ can obtain. Then the second-level optimisation problem is equivalent to another set packing problem on the remaining objects. The bi-level problem becomes:

$$\begin{aligned} \max_{\boldsymbol{x}_{\mathcal{I}}} \ & \boldsymbol{v}_{\mathcal{I}}^t\boldsymbol{x}_{\mathcal{I}} \ + f_{IP}^*(\mathcal{N}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{v}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}), \\ & \boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}} \leq \boldsymbol{e}, \ \boldsymbol{x}_{\mathcal{I}} \in \{0,1\}^{|\mathcal{I}|}. \end{aligned} \tag{4}$$

It is interesting to note that the reformulation has a smaller number of binary variables, i.e. $\boldsymbol{x}_{\mathcal{I}} \in \{0,1\}^{|I|}$, instead of 2^n binary variables that appear in model (1). However, this comes at a cost of having an unknown non-linear term $f_{IP}^*(\mathcal{N}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{v}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}})$. Indeed, this quantity is the payoff of the optimal set packing problem on the remaining objects. This means the problem is still hard but this makes sense since two equivalent formulations should have the same level of complexity. Notice, however, that the reformulation can provide us an idea of how to find a near-optimal solution as will be described next.

2.2. Finding Near-Optimal Set Packings (Lower Bounds)

Instead of maximising the entire quantity $\{\boldsymbol{v}_{\mathcal{I}}^t\boldsymbol{x}_{\mathcal{I}} + f_{IP}^*(\mathcal{N}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{v}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}})\}$, let us approximate the problem by replacing the second term $f_{IP}^*(\mathcal{N}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{v}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}})$ with its linear approximation $\boldsymbol{c}^t(\boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}})$ as follows:

$$\begin{aligned} \mathbf{CSPP_{sub}} := \max_{\boldsymbol{x}} \ & \boldsymbol{v}_{\mathcal{I}}^t\boldsymbol{x}_{\mathcal{I}} + \boldsymbol{c}^t(\boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}), \\ s.t. \ & \boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}} \leq \boldsymbol{e}, \\ & \boldsymbol{x} \in \{0,1\}^{|I|}, \end{aligned}$$

where $\boldsymbol{c}$ is the vector of payoffs that individual objects can obtain and $(\boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}})$ is the indicator vector of the remaining objects. The corresponding formulation is an MILP with n linear constraints and with $|\mathcal{I}|$ variables and could be solved efficiently by CPLEX, or some existing algorithms for solving the set packing problem (e.g. [6, 10, 19, 32]) for relatively small $|\mathcal{I}|$. We notice that the objective function is equivalent to $(\boldsymbol{v}_{\mathcal{I}}^t - \boldsymbol{c}^t\boldsymbol{A}_{\mathcal{I}})\boldsymbol{x}_{\mathcal{I}}$ where $(\boldsymbol{v}_{\mathcal{I}}^t - \boldsymbol{c}^t\boldsymbol{A}_{\mathcal{I}})$ is the corresponding reduced cost vector for $\boldsymbol{x}_{\mathcal{I}}$. After having found $\boldsymbol{x}_{\mathcal{I}}^*$, we can solve a new CSPP problem on the remaining objects to find $f_{IP}^*(\mathcal{N}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}^*, \boldsymbol{v}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}^*)$. This can be done through an exact algorithm if the number of remaining objects is relatively small, or through an approximation method such as the one we are describing otherwise. Formally, this is described in Algorithm 1.

In step (**1**) of Algorithm 1, we find an initial set of potential subsets that might appear in the optimal packing. We will use the set $\mathcal{I}$ that is suggested by the LP relaxation solution. A possible method for solving the LP relaxation is described in subsection 2.4. Notice, however, that the algorithm is very flexible in choosing the candidate set. In fact, the set $\mathcal{I}$ can also be obtained from any heuristic solution or from a combination thereof. In step (**2**) we solve problem $\mathbf{CSPP_{sub}}$ to find a near-optimal solution to the reformulation of CSPP in model (4). In step (**3**) we update the

Algorithm 1 Approximation Algorithm for Finding Near-Optimal Set Packing (Lower Bound)

Initialisation: Set $k=1$, $\mathcal{N}^{(k)}=\mathcal{N}$, $f^*_{sub}=0$.

while $\mathcal{N}^{(k)} \neq \emptyset$ **do**

1. Find the set of candidate subsets $\mathcal{I}$ that is suggested by an LP relaxation solution.

2. Solve $\mathbf{CSPP_{sub}}$. Let $\boldsymbol{x}^*_{\mathcal{I}}$ be an optimal solution.

3. Update:

$$f^*_{sub}=f^*_{sub}+\boldsymbol{v}^t_{\mathcal{I}}\boldsymbol{x}^*_{\mathcal{I}},\ N^{(k+1)}=N^{(k)}\backslash \boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{v}^{(k+1)}=\boldsymbol{v}^{(k)}\backslash \boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}},\ k=k+1.$$

end while

Return solution f^*_{sub}.

approximated objective value, the set of remaining objects, and their reduced vector of the payoff values. The first result we can see immediately is that the algorithm provides us a feasible packing solution and a lower bound to the CSPP as formally stated in the Lemma 1.

LEMMA 1. *Algorithm 1 produces a lower bound on the CSPP, i.e.* $f^*_{sub} \leq f^*_{IP}$.

Proof Since Algorithm 1 selects no same object twice for different subsets, it produces a feasible packing solution with f^*_{sub} being the packing's total payoff. Thus, $f^*_{sub} \leq f^*_{IP}$. □

Given the lower bound of CSPP, we are interested in judging how good the bound is. In theory, the quality of the approximated solution depends on how good the linear approximation $\boldsymbol{c}^t(\boldsymbol{e}-\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}})$ is compared to $f^*_{IP}(\mathcal{N}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{v}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}})$ around the truth optimum. In general, the linear approximation is more accurate if $||\boldsymbol{e}-\boldsymbol{A}\boldsymbol{x}_{\mathcal{I}}||$ is small. In other words, the higher the accuracy of picking the right candidate subset $\mathcal{I}$, the smaller the size of the remaining set, and hence the more accurate the linear approximation is. It is important that the approximated function $\tilde{f}(\boldsymbol{x}_{\mathcal{I}})$ is close to the original objective function $f(\boldsymbol{x}_{\mathcal{I}})$ around the optimum $\boldsymbol{x}_{\mathcal{I}}$. Notice also that, due to the discreteness of $\boldsymbol{x}_{\mathcal{I}}$, it is still possible for $\tilde{f}(\boldsymbol{x}_{\mathcal{I}})$ to have the same optimum as $f(\boldsymbol{x}_{\mathcal{I}})$ even though the two functions differ at $\boldsymbol{x}^*_{\mathcal{I}}$. It is easy to extend the algorithm to use a quadratic approximation of $f^*_{IP}(\mathcal{N}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}}, \boldsymbol{v}\backslash\boldsymbol{A}_{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}})$ and this would enhance the quality of the bound. However, this will come at a cost of having a more computationally expensive approximation and hence the choice would depend on how we want to trade off between the quality of the solution and the computation time. We will use the upper bounds found via LP and MILP relaxation (to be described in subsections 2.3 and 2.4) to estimate the optimality gap of the solutions found by Algorithm 1.

2.3. Finding Upper Bound via MILP Relaxation

In LP relaxation, we relax the binary constraints on $\boldsymbol{x}$ and this provides us with an upper bound to the CSPP. However, due to the excessive relaxation on all the 2^n binary variables, the quality of the bound might not be good enough for some instances. With the expectation that the optimal

columns are mostly in $\boldsymbol{A}_{\mathcal{I}}$, we develop a better relaxation strategy where only the vector $\boldsymbol{x}_{\mathcal{J}}$ is relaxed while still enforcing $\boldsymbol{x}_{\mathcal{I}}$ to be binary as follows:

$$\begin{aligned}\mathbf{CSPP_{MIP}} := \max_{\boldsymbol{x}} \quad & \boldsymbol{v}_{\mathcal{I}}^t \boldsymbol{x}_{\mathcal{I}} + \boldsymbol{v}_{\mathcal{J}}^t \boldsymbol{x}_{\mathcal{J}}, \\ s.t. \quad & \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}} + \boldsymbol{A}_{\mathcal{J}} \boldsymbol{x}_{\mathcal{J}} \leq \boldsymbol{e}, \\ & \boldsymbol{x}_I \in \{0,1\}^{|I|}, \\ & \boldsymbol{x}_{\mathcal{J}} \geq 0,\end{aligned}$$

where the constraint $\boldsymbol{x}_J \in \{0,1\}^{|J|}$ has been relaxed to $\boldsymbol{x}_{\mathcal{J}} \geq 0$. Notice that we do not have to include $\boldsymbol{x}_{\mathcal{J}} \leq 1$ because the packing constraint $\{\boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}} + \boldsymbol{A}_{\mathcal{J}} \boldsymbol{x}_{\mathcal{J}} \leq \boldsymbol{e}\}$ has already enforced this. We call this an MIP relaxation and notice that it has $|\mathcal{I}|$ binary variables and $|\mathcal{J}|$ continuous variables. We will choose $|\mathcal{I}|$ relatively small, i.e. $|\mathcal{I}| \sim n \ll 2^n$, and hence $\mathbf{CSPP_{MIP}}$ is much easier to solve compared to the original MILP but still more difficult than the LP relaxation due to the presence of the binary variable $\boldsymbol{x}_{\mathcal{I}}$. Solving $\mathbf{CSPP_{MIP}}$ will provide us a better upper bound to the CSPP compared to the LP relaxation, as stated in the following lemma.

LEMMA 2. *The following inequalities hold:* $f_{LP}^* \geq f_{MIP}^* \geq f_{IP}^* \geq f_{sub}^*$.

Proof The first two inequalities are obvious since $\mathbf{CSPP_{LP}}$ is a linear relaxation of $\mathbf{CSPP_{MIP}}$ (on variable $\boldsymbol{x}_{\mathcal{I}}$) and $\mathbf{CSPP_{MIP}}$ is a linear relaxation of **CSPP** (on variable $\boldsymbol{x}_{\mathcal{J}}$). The last inequality was derived in Lemma 1. □

Despite the tighter bound obtained, solving $\mathbf{CSPP_{MIP}}$ given its exponential size is very challenging. Due to the presence of the binary variable $\boldsymbol{x}_{\mathcal{I}}$, we cannot apply the classical column generation approach to handle the exponentially large number of columns. In what follows, we will present a new approach for solving $\mathbf{CSPP_{MIP}}$.

We notice that $\mathbf{CSPP_{MIP}}$ can be reformulated as a bi-level optimisation problem as follows:

$$\max_{\boldsymbol{x}_{\mathcal{I}} \in \{0,1\}^{|\mathcal{I}|}} \boldsymbol{v}_{\mathcal{I}}^t \boldsymbol{x}_{\mathcal{I}} + \max_{\boldsymbol{x}_{\mathcal{J}} \geq 0} \boldsymbol{v}_{\mathcal{J}}^t \boldsymbol{x}_{\mathcal{J}}, \quad \boldsymbol{A}_{\mathcal{J}} \boldsymbol{x}_{\mathcal{J}} \leq \boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}}, \quad \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}} \leq \boldsymbol{e}.$$

Due to the boundedness and the non-emptiness of the constraint set on $\boldsymbol{x}_{\mathcal{J}}$, we can replace the LP in the second level with its dual:

$$\max_{\boldsymbol{x}_{\mathcal{I}} \in \{0,1\}^{|\mathcal{I}|}} \boldsymbol{v}_{\mathcal{I}}^t \boldsymbol{x}_{\mathcal{I}} + \min_{\boldsymbol{\beta} \geq 0} \boldsymbol{\beta}^t (\boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}}), \quad \boldsymbol{A}_{\mathcal{J}}^t \boldsymbol{\beta} \geq \boldsymbol{v}_{\mathcal{J}}, \quad \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}} \leq \boldsymbol{e}.$$

Let $\mathcal{F}_J := \{\boldsymbol{\beta} : \boldsymbol{A}_{\mathcal{J}}^t \boldsymbol{\beta} \geq \boldsymbol{v}_{\mathcal{J}}, \boldsymbol{\beta} \geq 0\}$; the bi-level problem can be further reformulated as:

$$\begin{aligned}\max_{\boldsymbol{x}_I \in \{0,1\}^{|I|}, \delta} \quad & \boldsymbol{v}_{\mathcal{I}}^t \boldsymbol{x}_{\mathcal{I}} + \delta \\ s.t. \quad & \boldsymbol{\beta}^t (\boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}}) \geq \delta \ \forall \boldsymbol{\beta} \in \mathcal{F}_J.\end{aligned} \tag{5}$$

This reformulation is an MIP with $|\mathcal{I}|$ binary variables and one continuous variable, and with a semi-infinite number of constraints (one for each $\boldsymbol{\beta} \in \mathcal{F}_J$). Due to the linearity of the constraints on $\boldsymbol{\beta}$, the semi-infinite constraints are equivalent to a finite set of constraints for all the extreme points and extreme rays $\{\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^K\}$ of $\mathcal{F}_J$. Model (5) can be reformulated as:

$$\begin{aligned} \mathbf{CSPP_{MIP2}} := \max_{\boldsymbol{x}_I \in \{0,1\}^{|I|}, \delta} \quad & \boldsymbol{v}_{\mathcal{I}}^t \boldsymbol{x}_{\mathcal{I}} + \delta \\ s.t. \quad & \boldsymbol{\beta}^{(k)t}(\boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}}) \geq \delta, \ \forall k = 1, \ldots, K. \end{aligned} \tag{6}$$

The new MIP problem has $|\mathcal{I}|$ binary variables, one continuous variable, and K linear constraints where K could be exponentially large. This means the reformulation is still as complex as the original MIP but this makes sense since the two are equivalent. However, the formulation defined by $\mathbf{CSPP_{MIP2}}$ allows us to apply a constraint generation framework where only relevant extreme points and extreme rays of $\mathcal{F}_J$ are introduced to the model, as described in Algorithm 2.

Algorithm 2 Constraint Generation Algorithm for Solving MIP relaxation

Initialisation: Start with any initial relaxed constraint set $\mathcal{M}^{(0)} = \{\boldsymbol{\beta}^{(0)}\}$, where $\boldsymbol{\beta}^{(0)}$ is a feasible point in $\mathcal{F}_J$ and set $k = 0$.

loop

1. Set $k = k + 1$ and solve the **relaxed problem**:

$$(\boldsymbol{x}_I^{(k)}, \delta^{(k)}) \ = \ \arg\min_{\boldsymbol{x}, \delta} \left\{ \boldsymbol{v}_{\mathcal{I}}^t \boldsymbol{x}_{\mathcal{I}} + \delta \ : \ (\boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}})^t \boldsymbol{\beta}^{(j)} \geq \delta, \ \forall j \in \{0, \ldots, k-1\} \right\}.$$

2. Solve the **constraint generation problem**:

$$\boldsymbol{\beta}^{(k)} \ = \ \min_{\boldsymbol{\beta}} \left\{ \boldsymbol{\beta}^t (\boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}}^{(k)}) \ : \ \boldsymbol{A}_{\mathcal{J}}^t \boldsymbol{\beta} \geq \boldsymbol{v}_{\mathcal{J}}, \ \boldsymbol{\beta} \geq 0. \right\}$$

if $(\boldsymbol{e} - \boldsymbol{A}_{\mathcal{I}} \boldsymbol{x}_{\mathcal{I}}^{(k)})^t \boldsymbol{\beta}^{(k)} \geq \delta^{(k)}$ **then**

Terminate the loop.

else

Update $\mathcal{M}^{(k)} = \{\mathcal{M}^{(k-1)}, \boldsymbol{\beta}^{(k)}\}$.

end if

end loop

Return optimal solution $\boldsymbol{x}_I^* = \boldsymbol{x}_I^{(k)}$ and set $f_{MIP}^* = \boldsymbol{v}_{\mathcal{I}}^t \boldsymbol{x}_{\mathcal{I}}^* + \delta^{(k)}$.

In step (**1**) of Algorithm 2, we solve a relaxed problem of $\mathbf{CSPP_{MIP2}}$. This is an MIP and can be handled by CPLEX for reasonably large n (up to a few hundred objects) as long as we keep track of the number of constraints generated, i.e. we keep k below some bound relative to n. In step (**2**) we solve the constraint generation problem. Notice the problem has exactly the same structure compared to the dual of $\mathbf{CSPP_{LP}}$. We describe a solution approach for solving this problem in subsection 2.4.

It is important to note that we cannot apply a column generation algorithm directly to the original MIP as that would not provide us a proof of optimality even when no further improving column can be found, i.e. we could be trapped in local optimum due to the presence of the binary variable $\boldsymbol{x}_{\mathcal{I}}$. However, with the reformulation in $\mathbf{CSPP_{MIP2}}$, we have a proof of optimality on the constraint generation algorithm once no further violating constraints are found.

2.4. Solving the LP Dual

Consider the dual of $\mathbf{CSPP_{LP}}$:

$$\begin{aligned} \mathbf{CSPP_{LPD}} := \min \ & \boldsymbol{e}^t\boldsymbol{y}, \\ s.t. \ & \boldsymbol{a}_j^t\boldsymbol{y} \geq v(\boldsymbol{a}_j), \ \forall j \in \{1,..,2^n\}, \\ & \boldsymbol{y} \geq 0. \end{aligned}$$

The dual problem contains a decision variable $\boldsymbol{y} \in \mathbb{R}^n$ and an exponentially large number of constraints. The optimal value of the dual LP relaxation problem, denoted as f^*_{LP}, provides us an upper bound on the optimal value of the original MILP. From this, we obtain f^*_{sub}/f^*_{LP} as a guaranteed optimality bound of the feasible packing found in subsection 2.2. Although this upper bound is not as tight as the MIP upper bound, it is computationally less expensive and might be more appropriate to use in situations with limited computational resources. For $n \leq 20$, we can solve $\mathbf{CSPP_{LPD}}$ easily by using an LP solver like CPLEX. For $n \geq 30$, the problem contains more than a trillion constraints and this makes it difficult to solve. However, it is interesting to observe that only a small number of constraints are tight at the optimal solution. This means the remaining non-binding constraints can be relaxed without changing the optimal solution. We can start with a relaxed problem of $\mathbf{CSPP_{LPD}}$ with a small set of constraints and then keep introducing violating constraints to the relaxed problem until all the constraints are satisfied. At that point, the optimal solution of the relaxed problem is also the optimal solution of the original problem. Formally, the constraint generation method for solving $\mathbf{CSPP_{LPD}}$ is described in Algorithm 3.

In step (**1**) of Algorithm 3, we find a subset $\boldsymbol{z}$ that violates constraint $\{\boldsymbol{z}^t\boldsymbol{y}^{(k)} \geq v(\boldsymbol{z})\}$ mostly for the given proposal $\boldsymbol{y}^{(k)}$. Here, a subset is characterised by a binary indicator vector $\boldsymbol{z}$ with $z_i = 1$ if object i is in the subset and $z_i = 0$ otherwise. We then check the optimality condition. If the worst subset is not violated, then all other subsets satisfy this constraint and hence $\boldsymbol{y}^{(k)}$ is an optimal solution of $\mathbf{CSPP_{LPD}}$. Otherwise, we introduce the newly generated constraint $\{\boldsymbol{w}^t\boldsymbol{y} - v(\boldsymbol{w}) \geq 0\}$ to the relaxed problem. In step (**2**), we solve the updated relaxed problem to obtain a new proposal $\boldsymbol{y}^{(k)}$ before going back to step (**1**).

Notice that the relaxed problem is an LP with smaller size and is easy to solve. The key, and often the difficult part, for a successful constraint generation algorithm is the ability to generate violating constraints efficiently. We will show that the CG problem in the spectrum auction application can be solved in $\mathrm{O}(n^2 \log n)$. We will also show that the constraint generation problem in the WCSG

Algorithm 3 Constraint Generation Algorithm for Solving LP relaxation

Initialisation: Start with any initial weight vector $\boldsymbol{y}^{(0)}$ and initial relaxed constraint set $\mathcal{M}^{(0)}$ and set $k=0$.

loop

1. Solve the **constraint generation problem**: $\boldsymbol{w} = \underset{\boldsymbol{z}\in\{0,1\}^n}{\arg\min} \left\{\boldsymbol{z}^t\boldsymbol{y}^{(k)} - v(\boldsymbol{z})\right\}$.

if $\boldsymbol{w}^t\boldsymbol{y}^{(k)} - v(\boldsymbol{w}) \geq 0$ **then**

Terminate the loop.

else

Update relax set: $\mathcal{M}^{(k+1)} = \{\mathcal{M}^{(k)}, \boldsymbol{w}\}$.

end if

2. Set $k = k+1$ and solve the **relaxed problem**: $\boldsymbol{y}^{(k)} = \underset{\boldsymbol{y}}{\arg\min} \left\{\boldsymbol{e}^t\boldsymbol{y} \ : \ \boldsymbol{z}^t\boldsymbol{y} \geq v(\boldsymbol{z}),\ \forall \boldsymbol{z} \in \mathcal{M}^{(k)}\right\}$.

end loop

Return optimal solution $\boldsymbol{y}^* = \boldsymbol{y}^{(k)}$ and set $f^*_{LP} = \boldsymbol{e}^t\boldsymbol{y}^*$.

games with reasonable size (i.e. $n \leq 200$) can also be solved very efficiently. Some remarks on the constraint generation problem follow.

- The constraint generation problem does not have to be solved exactly at every step except for the last step. In fact, any violating constraint could be added to the relaxed problem. It is only in the last step before terminating the loop that we need to obtain an optimal solution $\boldsymbol{w}$ and to check if $\{\boldsymbol{w}^t\boldsymbol{y}^{(k)} - v(\boldsymbol{w}) \geq 0\}$.
- In some instances, the constraint generation problem might turn out to be very difficult to be solved to optimality. In this case, we can still obtain an upper bound to the CSPP as stated in Theorem 1.

THEOREM 1. *Let $y^{(k)}$ be the optimal solution of the relaxed problem at iteration k in Algorithm 3 and let $\epsilon = \max_j\{v_j - \boldsymbol{a}_j^t\boldsymbol{y}^{(k)}\}$; then we have:*

$$\boldsymbol{e}^t\boldsymbol{y}^{(k)} \leq \boldsymbol{e}^t\boldsymbol{y}^* \leq \boldsymbol{e}^t\boldsymbol{y}^{(k)} + n\epsilon.$$

Proof Since $\boldsymbol{y}^*$ is an optimal solution of $\mathbf{CSPP_{LPD}}$, it is also a feasible solution of any LP relaxation problem at any iteration k. Thus, its objective value $\boldsymbol{e}^t\boldsymbol{y}^*$ should be at least the optimal value $\boldsymbol{e}^t\boldsymbol{y}^{(k)}$ of the relaxed LP, i.e. $\boldsymbol{e}^t\boldsymbol{y}^{(k)} \leq \boldsymbol{e}^t\boldsymbol{y}^*$. We can also show that $\boldsymbol{x} = \boldsymbol{y}^{(k)} + \epsilon\boldsymbol{e}$ is a feasible solution to the relaxed LP by verifying that both the constraints $\boldsymbol{a}_j^t\boldsymbol{x} \geq v(\boldsymbol{a}_j),\ \forall j \in \{1,..,2^n\}$ and $\boldsymbol{x} \geq 0$ are satisfied. By the definition of $\epsilon = \max_j\{v_j - a_j^t\boldsymbol{y}^{(k)}\}$ and by the construction of the algorithm, we have $\epsilon \geq 0$ before the loop in Algorithm 3 terminates. Thus, $\boldsymbol{x} = \boldsymbol{y}^{(k)} + \epsilon\boldsymbol{e} \geq \boldsymbol{y}^{(k)} \geq 0$. For $j=1$ we have $\boldsymbol{a}_j = 0$ and $v_j = 0$ and hence the constraint $\boldsymbol{a}_j^t\boldsymbol{x} \geq v(\boldsymbol{a}_j)$ holds trivially. For $j \geq 2$, we have:

$$\boldsymbol{a}_j^t\boldsymbol{x} = \boldsymbol{a}_j^t\boldsymbol{y}^{(k)} + \epsilon\boldsymbol{a}_j^t\boldsymbol{e}$$

$$\geq \boldsymbol{a}_j^t \boldsymbol{y}^{(k)} + \epsilon$$
$$\geq \boldsymbol{a}_j^t \boldsymbol{y}^{(k)} + (v_j - \boldsymbol{a}_j^t \boldsymbol{y}^{(k)}) = v_j.$$

Thus, $\boldsymbol{x}$ is a feasible solution to $\mathbf{CSPP_{LPD}}$. Therefore,

$$\boldsymbol{e}^t \boldsymbol{y}^* \leq \boldsymbol{e}^t \boldsymbol{x} = \boldsymbol{e}^t \boldsymbol{y}^{(k)} + \epsilon \boldsymbol{e}^t \boldsymbol{e} = \boldsymbol{e}^t \boldsymbol{y}^{(k)} + n\epsilon.$$

□

The implication of Theorem 1 is that if ϵ is sufficiently small relative to $\boldsymbol{e}^t \boldsymbol{y}^{(k)}$, we can stop the CG algorithm and use the upper bound $(\boldsymbol{e}^t \boldsymbol{y}^{(k)} + n\epsilon)$ instead of $\boldsymbol{e}^t \boldsymbol{y}^*$.

3. Applications to Combinatorial Auctions and Cooperative Game Theory

3.1. The Winner Determination Problem in Spectrum Auctions

In a spectrum auction, there are n frequencies (objects) and m bidders. Each bidder is interested in a subset of frequencies. Let $\Phi \in \mathbb{R}^{m \times n}$ be a 0-1 matrix where $\phi_{ij} = 1$ indicates whether bidder i has an interest in frequency j and $\phi_{ij} = 0$ otherwise. We follow the model suggested by Cramton et al. [13] and De Vries and Vohra [15] and assume that the individual object j has a payoff value of v_j. Bidder i will value a subset $\mathcal{S}$ of frequencies as $\{\sum_{i \in \mathcal{S}} v_i + \mu_i \sum_{k,q \in \mathcal{S} \cap \boldsymbol{\phi}_i} v_k v_q\}$ where μ_i is a parameter that models how strong bidder i views the complementarities (De Vries and Vohra [15]). The largest bid on a subset $\mathcal{S}$ is:

$$v(\boldsymbol{z}) = \max_{i \in \mathcal{N}} \left\{ \sum_{i \in \mathcal{S}} v_i + \mu_i \sum_{k,q \in \mathcal{S} \cap \boldsymbol{\phi}_i} v_k v_q \right\}. \tag{7}$$

Notice that $v(\boldsymbol{z})$ is not given explicitly. To apply existing methods, we would have had to evaluate $v(\boldsymbol{z})$ for all $\boldsymbol{z}$ just to obtain the input to the CSPP. This is very expensive computationally. We will show that, by applying our constraint generation framework, we will eliminate this stage and only perform the calculation if needed. Specifically, we can combine the max operator in (7) into the max operator of the CG problem as follows:

$$\begin{aligned}
&\max_{\boldsymbol{z}} \ \left\{ v(\boldsymbol{z}) - \boldsymbol{z}^t \boldsymbol{y} \right\} \\
\Leftrightarrow &\max_{\boldsymbol{z}, i \in \mathcal{N}} \ \left\{ \sum_{i \in \mathcal{S}} v_i + \mu_i \sum_{k,q \in \mathcal{S} \cap \boldsymbol{\phi}_i} v_k v_q - \boldsymbol{z}^t \boldsymbol{y} \right\} \\
\Leftrightarrow &\max_{i \in \mathcal{N}} \ \max_{\boldsymbol{z}} \ \left\{ \sum_{i \in \mathcal{S}} v_i + \mu_i \sum_{k,q \in \mathcal{S} \cap \boldsymbol{\phi}_i} v_k v_q - \boldsymbol{z}^t \boldsymbol{y} \right\} \\
\Leftrightarrow &\max_{i \in \mathcal{N}} \ \max_{\boldsymbol{z}} \ \left\{ \boldsymbol{v}^t \boldsymbol{z} - \boldsymbol{z}^t \boldsymbol{y} + \mu_i ((\boldsymbol{\phi}_i \odot \boldsymbol{v})^t \boldsymbol{z})^2 \right\} \\
\Leftrightarrow &\max_{i \in \mathcal{N}} \ \max_{\boldsymbol{z}} \ \left\{ (\boldsymbol{b}^t \boldsymbol{z})^2 - \boldsymbol{a}^t \boldsymbol{z} \right\},
\end{aligned} \tag{8}$$

where $\boldsymbol{a} = \boldsymbol{y} - \boldsymbol{v}$, and $\boldsymbol{b} = (\boldsymbol{\phi}_i \odot \boldsymbol{v})$ is the element-wise production of the two vectors $\boldsymbol{\phi}_i$ and $\boldsymbol{v}$.

For each fixed i, the CG problem is a quadratic binary optimisation problem (QBP) which is NP-hard to solve except for special cases. Among these, Allemand et al. [1] develop a polynomial time algorithm for solving unconstrained fixed-ranked homogeneous QBP, i.e. one with the form $\max_{\boldsymbol{z}\in\{0,1\}^n} \{\boldsymbol{z}^t\boldsymbol{Q}\boldsymbol{z}\}$ where $\boldsymbol{Q}$ is a symmetric positive definite matrix with a fixed rank. Specifically, the authors show that there is an $\mathrm{O}(n^{d-1})$ algorithm for solving the QBP where d is the rank of $\boldsymbol{Q}$. We will extend this result to the case of nonhomogeneous fixed-ranked QBP and show an $\mathrm{O}(n^2 \log n)$ algorithm for solving the constraint generation problem.

THEOREM 2. *The constraint generation problem described in (8) can be solved in $O(n^2 \log n)$.*

Proof In the CG problem, we need to maximise $\{(\boldsymbol{b}^t\boldsymbol{z})^2 - \boldsymbol{a}^t\boldsymbol{z}\}$. For each $\boldsymbol{z} \in \{0,1\}^n$, let $P(\boldsymbol{z}) = (\boldsymbol{a}^t\boldsymbol{z}, \boldsymbol{b}^t\boldsymbol{z})$ be the corresponding point in a 2-dimensional space, i.e. $P(\boldsymbol{z}) : \{0,1\}^n \to \mathbb{R}^2$. Since there are 2^n such points $\boldsymbol{z}$, we have 2^n corresponding points $P(\boldsymbol{z})$ in 2-D. It is very interesting, however, that the convex hall of these points has at most $2n$ extreme points (Allemand et al. [1]). Since we are maximising a convex function $\{P_2(\boldsymbol{z})^2 - P_1(\boldsymbol{z})\}$, the maximum is attained at one of the extreme points. In addition, these extreme points can be listed out as follows:

Let $p_i = (a_i, b_i)$ and let $\alpha_i = \arctan(a_i/b_i)$. We first order α in an ascending order $\alpha_{(1)} \le \alpha_{(2)} \ldots \le \alpha_{(n)}$. The $2n$ extreme points are given by the following equations:

$$E_k = \begin{cases} \mathbf{0} & \text{if } k = 0 \\ \sum_{j \le k} p_{(j)} & \text{if } 1 \le k \le n \\ \sum_{j=k+1-n}^{n} p_{(j)} & \text{if } n+1 \le k \le 2n-1. \end{cases}$$

Once these points have been found, we can compare the objective values $\{E_2(\boldsymbol{z})^2 - E_1(\boldsymbol{z})\}$ only among these $2n$ extreme points to find the maximum. This takes $\mathrm{O}(n)$ operations and the sorting of vector α takes $n\log(n)$ operations. Since we have to solve this problem n times, one for each bidder, the complexity of the algorithm is $\mathrm{O}(n^2\log(n))$. □

Figure 1 demonstrates the algorithm for solving the CG problem for the case of $n = 10$ for some fixed i. Crossed generator points $p = (p_1, p_2)$ are generated randomly with p_1 following a uniform distribution in $[0,1]$ while $p_2 = \sqrt{0.2u}$ with u also following a uniform distribution. These choices of these random points are only for the purpose of clearer demonstration in Figure 1. The corresponding zonotope has 20 extreme points. The maximum of $\{P_2^2 - P_1\}$ is attained at E_{13}. This means $z_{(j)} = 0$, for $1 \le j \le 3$ and $z_{(j)} = 1$ for $4 \le j \le 10$.

THEOREM 3. *The LP dual problem* $\mathbf{CSPP_{LPD}}$ *in spectrum auction can be solved in polynomial time.*

Proof From Theorem 2, identifying a violating constraint for a given $\boldsymbol{y}$ can be done in $\mathrm{O}(n^2 \log n)$. This means we have an 'oracle-polynomial time' to solve the separation problem. Therefore, $\mathbf{CSPP_{LPD}}$ can be solved in polynomial time using the ellipsoid method (see Theorem 6.4.9 in Grotschel et al. [21]). □

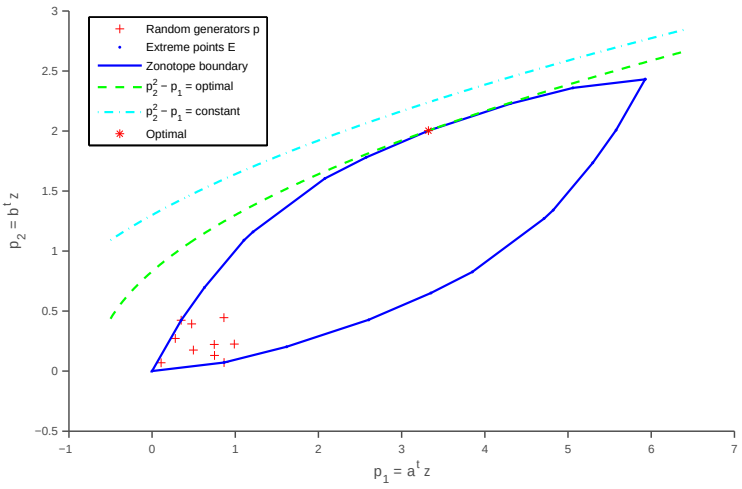

Figure 1 Zonotope demonstration for $n = 10$.

3.2. Finding the Optimal Coalition Structure in the WCSG games

The weighted coalitional skill games (WCSG) were proposed by Bachrach and Rosenschein [3]. In this game, there are n players, T tasks and K skills. Each player has a subset of skills and each task requires a subset of skills. Let Ψ be the player-skill matrix with ψ_{ik} indicating whether player i has skill k. Let Φ be the task-skill matrix with ϕ_{tk} indicating whether task t requires skill k. The players can form groups to work collectively and a group can perform a task if the skills required are available. Specifically, for each task t and the skill vector Φ_t that it requires, a coalition $\boldsymbol{z}$ will be able to perform the task if, for all skill k, there exists at least one player in the coalition that has skill ϕ_{tk}. Each task, if performed, returns some reward. The aim of the coalition structure generation problem in the WCSG game is to divide players into non-overlapping subgroups such that the total reward is maximised and this is equivalent to the CSPP problem. Bachrach and Rosenschein (2008) show that the problem is NP-hard except for some very restricted cases, such as with the bounded number of tasks and the bounded width of the tree that represent the skill-graph. In addition, even with these restrictions, the authors only provide a polynomial-time algorithm without any numerical result. We will apply our algorithm to find near-optimal CSPP of general WCSG games.

Let $\Delta(\boldsymbol{z}, t)$ be the binary indicator on whether coalition $\boldsymbol{z}$ can perform task t and it is defined as:

$$\Delta(\boldsymbol{z}, t) = \begin{cases} 1 & \text{if } \Psi^t \boldsymbol{z} \geq \Phi_t, \\ 0 & \text{otherwise.} \end{cases}$$

We consider a weighted average utility function defined as $v(\mathcal{S}) = \sum_{t \in \{1..T\}} \omega_t \Delta(\boldsymbol{z}, t)$. In this case, the constraint generation problem $\min_{\boldsymbol{z} \in \{0,1\}^n} \quad \{\boldsymbol{z}^t \boldsymbol{y}^{(k)} - v(\boldsymbol{z})\}$ can be reformulated as:

$$\min_{\boldsymbol{z} \in \{0,1\}^n} \left\{ \boldsymbol{z}^t \boldsymbol{y}^{(k)} - \sum_{t=1}^{T} \omega_t \Delta(\boldsymbol{z}, t) \right\}.$$

This is equivalent to:

$$\min_{\boldsymbol{z},\delta} \left\{ \boldsymbol{z}^t \boldsymbol{y}^{(k)} - \sum_{t=1}^{T} \omega_t \delta_t \right\}, \tag{9}$$

$$s.t. \quad \Psi^t \boldsymbol{z} \geq \delta_t \Phi_t, \ \forall t \in 1,..,T, \tag{10}$$

$$\boldsymbol{z} \in \{0,1\}^n, \psi_t \in \{0,1\}^m,$$

where $\Delta(\boldsymbol{z}, t)$ has been replaced by δ_t. The set of constraints in (10) ensures that if the coalition $\boldsymbol{z}$ does not have all the skills required in Φ_t, then δ_t must be equal to zero. Otherwise, δ_t should be equal to one to drive the objective function to the minimum. The constraint generation problem is a mixed-integer programming problem with $(n+T)$ binary variables and with $(T*K)$ constraints. Although the problem is NP-hard, we will show numerically that CPLEX can solve the game very efficiently for many instances with up to 200 players (with 20 tasks and 10 skills).

4. Numerical Tests

4.1. The Winner Determination Problem in Spectrum Auctions

We perform numerical tests on spectrum auctions with the number of frequencies ranging from $n = 25$ to $n = 200$. The number of bidders is set to be equal to the number of frequencies. Matrix Φ is generated randomly using the Bernoulli distribution where each bidder has a 50% chance of having an interest in a frequency. The payoff vector $\boldsymbol{v}$ and the synergy preference $\boldsymbol{\mu}$ are also generated randomly using the uniform distribution around $[0, 1]$.

Table 1 shows the performance of Algorithms 1-3 when the number of objects varies between 25 and 200. Columns 2-5 show statistics for solving the LP dual problem, columns 6-7 show statistics for finding upper bounds, and column 8 shows the optimality bounds. In each column, the statistics shown are obtained by taking the average over $K = 10$ random instances (using fixed random seeds between 1 and K in MATLAB for the purpose of convenient replication). By creating random instances, we can test the robustness of the algorithm when the input data vary.[1] In general, there is an increasing trend with a cubic shape in the total time taken to solve the LP dual problem except for the case of $n = 50$.[2] The total time taken to solve the LP dual is around 85 minutes for the largest instance. The number of iterations required by the constraint generation method to solve the LP dual is relatively small compared to the problem size. Specifically, to solve the LP dual with 2^{200} constraints for the case of $n = 200$, it took around 3273 iterations, on average, to identify the set of relevant constraints. This means we only need to solve around 3273 constraint generation problems and the same number of small-size LPs, each with $n = 200$ columns and

[1] All the numerical tests that appear in this manuscript are performed on a personal computer, Intel® Xeon® CPU W3520 @2.67GHz with 12GB RAM and under the Windows 7 operating system. The code was written and tested on MATLAB R2012a.

[2] Among 80 random instances generated, there were two outliers for the case of $n = 50$ with random seeds numbers 4 and 5.

with the number of rows ranging between 1 and 3273. The time required to solve the constraint generation problem increases almost in a quadratic trend and this matches with the theoretical complexity of $O(n^2 \log n)$ as derived in Theorem 2. The time taken to find feasible packings is relatively small compared to the time required to solve the LP relaxation with, at most, 20 minutes for the largest instance. The optimality bounds between the feasible solutions and the LP upper bounds are around 91.1%, which are reasonably good given the very large problem size. Notice that this is the guaranteed bounds taken by f^*_{sub}/f^*_{LP} as we have no knowledge of the optimal values. The actual optimality bounds could be higher.

	Solving LP dual				Finding lower bounds		Optimality bounds
n	Total time	Relaxed LPs	CG	# iterations	Total time	# iterations	f^*_{sub}/f^*_{LP}
25	4.92	0.87	3.77	215.6	0.10	3.2	97.46%
50	551.61	307.13	237.71	1437.5	1.82	3.9	94.24%
75	147.61	24.57	120.55	729.4	80.73	4.4	92.78%
100	265.85	46.89	215.38	1041	684.21	5	91.87%
125	526.54	129.23	391.27	1474.7	185.18	5.1	90.61%
150	966.3	289.23	667.92	1871.7	795.28	5.7	88.27%
175	2081.2	797.86	1267.8	2621.5	1202.3	6.2	86.94%
200	5060.1	2971.4	2064.2	3272.5	1005.9	6	86.42%

Table 1 Computational results for solving the winner determination problem in spectrum auctions (computational time is in seconds)

4.2. Optimal Coalition Structure Generation in the Coalitional Skill Games

We also perform numerical tests on large WCSG games with the number of players n ranging from 25 to 200 while fixing the number of tasks at 20 and the number of skills at 10. For each n, we generate $K = 10$ random samples using random seeds between 1 and K. In each instance, the player-skill matrix is generated randomly using the Bernoulli distribution with a 25% chance for each player to have a skill. The task-skill matrix is also generated randomly with a 50% chance that each task requires a skill. The weighted vector is generated uniformly between $[0, 1]$.

Table 2 shows the same statistics as Table 1 but for finding the optimal coalition structures of the WCSG with the number of players varying between 25 and 200 (i.e. columns 2-5 show statistics for solving the LP dual problem, columns 6-7 show statistics for finding upper bounds and column 8 shows the optimality bounds). In each column, the average performance over $K = 10$ runs are recorded. There is an increasing trend in the total time taken to solve the LP dual problem (with less than 10 minutes for the largest instance). Most of the time taken is for solving the constraint generation problems. The number of iterations to solve the LP dual is also relatively small with just over 300 iterations for the largest instances tested. The time required to find a feasible packing solution is less than 1 second and this is significantly small compared to the time taken to solve the LP dual problem. The optimality gap is around 99.5% which is quite impressive.

	Solving LP dual				Finding lower bounds		Optimality bounds
n	Total time	Relaxed LPs	CG	# iterations	Total time	# iterations	f^*_{sub}/f^*_{LP}
25	2.48	0.036	2.42	36.4	0.023	14.3	98.58%
50	13.32	0.086	13.14	65.8	0.037	28.3	99.46%
75	42.24	0.466	41.35	126.9	0.367	44	99.12%
100	83.34	0.541	81.94	162.6	0.513	56.3	99.46%
125	160.00	0.393	158.59	195.9	0.646	71.7	99.78%
150	245.93	0.764	243.60	276.8	0.655	84.8	99.81%
175	402.52	0.343	400.66	239.8	0.875	100	99.78%
200	487.50	0.846	484.54	303.4	0.846	114.4	99.73%

Table 2 Computational results for finding the optimal coalition structure of the weighted coalitional skill games (computational time is in seconds)

4.3. Random Instances from the CATS Library

The main aim of this manuscript is to find an approximation algorithm for solving large-scale SPP where the number of feasible subsets is very large, and hence cannot be handled by existing methods. Our method is, however, applicable to any SPP game with $m \gg n$. To demonstrate this, we test the algorithm with randomly generated problems that appear in the CATS library developed by Leyton-Brown and Shoham [27]. The value functions used in these instances have arisen in combinatorial auctions where the underlying bids are accompanied with economical interpretations. Tables 3 and 4 summarise the optimality bounds and the computational time for various random distributions shown in the first columns. Columns 2-5 show the optimality bounds for ($n = 100$) objects while varying the number of feasible subsets between 2500 and 10000. Columns 6-7 show the same statistics but for fixed ($m = 5000$) while varying n between 50 and 150. The rows show different distributions from the CATS library for generating random instances. For each distribution and each pair of (n, m), we generate $K = 10$ random instances. The statistics are taken from the average performance among these K instances. In general, it is interesting to see that the optimality bounds increase with the increase of m. The optimality bounds are 100%, i.e. we obtain optimal solutions, for the 'paths' and 'scheduling' distributions. The average is 99.3% for all cases. The computational time is less than 1 minute for the worst instance tested.

	n = 100				m = 5000		
Distributions	$m = 2500$	$m = 5000$	$m = 7500$	$m = 10000$	$n = 50$	$n = 100$	$n = 150$
Arbitrary	95.25%	96.86%	98.15%	99.28%	99.08%	96.86%	96.91%
Paths	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Matching	99.94%	99.98%	99.99%	99.99%	100.00%	99.98%	100.00%
Regions	98.10%	98.58%	99.36%	99.38%	99.66%	98.58%	98.45%
Scheduling	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Table 3 Optimality bounds f^*_{sub}/f^*_{LP} for various random instances in the CATS library

	n = 100				m = 5000		
Distributions	$m = 2500$	$m = 5000$	$m = 7500$	$m = 10000$	$n = 50$	$n = 100$	$n = 150$
Arbitrary	45.271	8.6534	6.7892	5.8766	1.7956	8.6534	210.85
Paths	0.35724	0.78781	1.2605	1.8112	0.70044	0.78781	0.93133
Matching	0.351	0.75504	1.2199	1.7831	0.81589	0.75504	0.81277
Regions	2.7628	2.6848	2.8923	3.3353	1.1435	2.6848	5.9795
Scheduling	0.34476	0.702	1.2262	1.8221	0.70356	0.702	0.72696

Table 4 Total computational time (in seconds) for various random instances in the CATS library

5. Conclusion

We provide an approximation method for solving the complete set packing problem (CSPP) where the family of feasible subsets might be exponentially large. This includes a new mathematical formulation with a much fewer number of binary variables that allows the direct pursuit of near-optimal solutions. We also develop an algorithm for solving a large-scale MIP to obtain tighter upper bounds to the CSPP. We show that the method works very effectively, both in computational time and the quality of the bounds obtained, with the applications in combinatorial auctions and in cooperative game theory. Specifically, we show that recent results in fixed-rank quadratic binary programming can be extended to prove that the LP relaxation problem in spectrum auctions can be solved in polynomial time. The fast performance of the constraint generation algorithm for solving the LP relaxation problem of the coalitional skill games implies that the core and the least core in these games can also be computed efficiently by using the same method. Results from this research, particularly the new formulation for the CSPP, may stimulate future research to find alternative methods for solving large-scale SPP problems. Our algorithm for solving the large-scale MIP could also be applicable to other applications that involve an exponentially large number of continuous variables.

References

[1] Kim Allemand, Komei Fukuda, Thomas M Liebling, and Erich Steiner. A polynomial case of unconstrained zero-one quadratic optimization. *Mathematical programming*, 91(1):49–52, 2001.

[2] Y. Bachrach, R. Meir, K. Jung, and P. Kohli. Coalitional structure generation in skill games. *AAAI-2010*, 2010.

[3] Yoram Bachrach and Jeffrey S Rosenschein. Coalitional skill games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2*, AAMAS '08, pages 1023–1030, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-0-9817381-1-6. URL `http://dl.acm.org/citation.cfm?id=1402298.1402364`.

[4] John E Beasley. An algorithm for set covering problem. *European Journal of Operational Research*, 31 (1):85–93, 1987.

[5] John E Beasley and Paul C Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392–404, 1996.

[6] John E Beasley and K Jörnsten. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58(2):293–300, 1992.

[7] Lázaro Cánovas, Mercedes Landete, and Alfredo Marın. New facets for the set packing polytope. *Operations Research Letters*, 27(4):153–161, 2000.

[8] Estelle Cantillon and Martin Pesendorfer. Auctioning bus routes: The london experience. *Combinatorial auctions*, 22:573–592, 2006.

[9] Chris Caplice and Yossi Sheffi. Combinatorial auctions for truckload transportation. *Combinatorial auctions*, 21:539–572, 2006.

[10] Alberto Caprara, Paolo Toth, and Matteo Fischetti. Algorithms for the set covering problem. *Annals of Operations Research*, 98(1-4):353–371, 2000.

[11] Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.

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

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

[14] Viet Dung Dang, R. K. Dash, A. Rogers, and N. R. Jennings. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *Proceedings of The Twenty First National Conference on Artificial Intelligence (AAAI)*, pages 635–640, 2006.

[15] Sven De Vries and Rakesh V Vohra. Combinatorial auctions: A survey, 2000.

[16] Sven De Vries and Rakesh V Vohra. Combinatorial auctions: A survey. *INFORMS Journal on computing*, 15(3):284–309, 2003.

[17] Xavier Delorme, Xavier Gandibleux, and Joaquin Rodriguez. Grasp for set packing problems. *European Journal of Operational Research*, 153(3):564–580, 2004.

[18] Yuval Emek, Magnús M Halldórsson, Yishay Mansour, Boaz Patt-Shamir, Jaikumar Radhakrishnan, and Dror Rawitz. Online set packing. *SIAM Journal on Computing*, 41(4):728–746, 2012.

[19] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the Computational Complexity of Combinatorial Auctions: Optimal and Approximate Approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 548–553, Stockholm, Sweden, August 1999.

[20] SH Gow and L.C. Thomas. Interchange fees for bank atm networks. *Naval Research Logistics (NRL)*, 45(4):407–417, 1998.

[21] M. Grotschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimizations*. Springer-Verlag, 1993.

[22] Karla Hoffman and Manfred Padberg. Set covering, packing and partitioning problems. *Encyclopedia of Optimization*, pages 3482–3486, 2001.

[23] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[24] Mercedes Landete, Juan Francisco Monge, and Antonio M Rodríguez-Chía. Alternative formulations for the set packing problem and their application to the winner determination problem. *Annals of Operations Research*, pages 1–24, 2012.

[25] D. Leech. Computing power indices for large voting games. *Management Science*, pages 831–838, 2003.

[26] J. Lemaire. Cooperative game theory and its insurance applications. *ASTIN Bulletin*, 21(1):17–41, 1991.

[27] Kevin Leyton-Brown and Yoav Shoham. A test suite for combinatorial auctions. *Combinatorial auctions*, 18:451–478, 2006.

[28] Rudolf Müller. Tractable cases of the winner determination problem. *Combinatorial auctions*, 13: 319–336, 2006.

[29] Manfred W Padberg. On the facial structure of set packing polyhedra. *Mathematical programming*, 5 (1):199–215, 1973.

[30] Talal Rahwan, Tomasz Michalak, and Nicholas R. Jennings. A hybrid algorithm for coalition structure generation. In *Twenty Sixth Conference on Artificial Intelligence (AAAI-12)*, Toronto, Canada, 2012.

[31] Michael H Rothkopf, Aleksandar Pekeč, and Ronald M Harstad. Computationally manageable combinational auctions. *Management science*, 44(8):1131–1147, 1998.

[32] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1):1–54, 2002.

[33] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1–2):209–238, 1999.

[34] R Velásquez and M Teresa Melo. A set packing approach for scheduling elective surgical procedures. In *Operations Research Proceedings 2005*, pages 425–430. Springer, 2006.

[35] Peter J Zwaneveld, Leo G Kroon, and Stan PM Van Hoesel. Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128(1):14–33, 2001.