A Real-Time Solution for Underlay Coexistence with Channel Uncertainty

Shaoran Li, Yan Huang, Chengzhang Li,
Y. Thomas Hou, Wenjing Lou
Virginia Tech, Blacksburg, VA

Brian Jalaian, Stephen Russell,
Benjamin MacCall
U.S. Army Research Laboratory, Adelphi, MD

***Abstract*—Underlay coexistence is an effective mechanism to improve spectrum efficiency by having picocells coexist with macrocell on the same spectrum. Due to a lack of cooperation between the primary users (PUs) in the macrocell and secondary users (SUs) in the picocell, it is impossible to have complete knowledge of channel gains between them. Under such circumstance, chance-constrained programming (CCP) is shown to be the ideal optimization tool to address such uncertainty. However, solutions to CCP are computationally intensive and cannot meet 5G's timing requirement. To address this problem, we propose a novel scheduler called GUC (stands for GPU-based Underlay Coexistence) to find an approximate solution to CCP in real-time. The essence of GUC is to decompose the original optimization problem into a large number of small problems that are suitable for parallel computation on GPU platforms. Through extensive experiments, we show that GUC reduces the scheduling computation time by at least 10,000 times comparing to commercial solvers (on CPU) while achieving an average of 90% optimality.**

I. Introduction

Underlay coexistence is able to improve spectrum efficiency by allowing concurrent transmission of secondary users (SUs) on the same spectrum used by the primary users (PUs) [1]. To achieve harmonious coexistence, SUs should control their transmission powers such that the aggregated interference to each nearby PU is below a threshold. Under such a model, the interference channel gains from the SUs to the PUs are needed for power control. However, in the absence of feedback from the PUs, the SUs can only measure these interference channel gains based on channel reciprocity and known signals (e.g., pilots) whenever overhearing the PUs' transmission. As a result, accurate knowledge of instantaneous interference channel gains and their distributions are hardly available and one can at best obtain and work with their mean and covariance estimated through long term measurements.

Without knowledge of distributions, there are two possible approaches: worst-case optimization and chance-constrained programming (CCP). It is well-known that the performance of worst-case optimization is overly conservative. CCP resolves this issue by allowing occasional violation of the interference threshold as long as such violation is below a small (given) probability. CCP exploits the fact that in reality, such occasional violations are usually not fatal or even tolerable by the PUs due to inherent channel coding and human perception system [2], [3]. By allowing such occasional violations, CCP promises to achieve much higher efficiency in the presence of channel uncertainty.

In this paper, we employ CCP to study underlay coexistence between picocells (for SUs) and a macrocell (for PUs) on the same spectrum [4], [5]. There are several prior efforts employing CCP to study underlay problems (see, e.g., [6]–[10]). A common issue of these prior efforts is that the computation time of the proposed solutions is too large to satisfy 5G time resolution. Specifically, the time resolution for scheduling inside 5G small cells can be as low as 250 μs with 60 kHz subcarrier spacing [11], [12] while computation time in existing solutions [6]–[10] ranges from hundreds of milliseconds to tens of seconds. As a result, none of the existing solutions to CCP can meet the stringent timing requirement in 5G scheduling (250 μs).

From an implementation perspective, to achieve the real-time requirement in a pico BS, a scheduling algorithm must be implemented on a small size platform with sufficient computation resources. In recent years, GPU has emerged to be a promising solution due to its ability to reduce computation time through massive parallel computation. For instance, the authors in [13] showed that, by exploiting parallel computation on a GPU platform, it is possible to implement proportional fair (PF) scheduler with near-optimal performance while meeting 5G's timing requirement. However, due to several fundamental differences between these two problems (in terms of problem settings and mathematical structures), the approach in [13] cannot be used here.

In this paper, we propose and design a real-time picocell scheduler for underlay coexistence with channel uncertainty. Our main contributions are summarized as follows:

- We propose and implement the first solution to CCP for underlay coexistence that can meet timing requirement in 5G scheduling.
- Our proposed solution – GPU-based Underlay Coexistence (GUC), decomposes the original problem into a large number of subproblems by fixing binary decision variables and choosing a limited number of promising subproblems based on closed-form metrics and random sampling. For each subproblem, we propose a fast heuristic solution using decent starting points and scaling-based local search to find a feasible solution with a very small number of computation cycles.

- By implementing GUC on an off-the-shelf NVIDIA GPU, we demonstrate that GUC reduces computational time by at least 10,000 times compared to a commercial solver (Gurobi on CPU) while maintaining an average of 90% optimality.

II. System Model and Problem Formulation

Consider a two-tier cellular network where several picocells are deployed inside a macrocell based on underlay coexistence. Each picocell covers a scale of a residential unit (e.g., a house) [14]. Users in the macrocell and picocells are considered as PUs and SUs respectively. The neighboring picocells are using different fractions of the macrocell's spectrum in a non-overlapping fashion to avoid inter-cell interference, which is known as fractional frequency reuse (see, e.g., [15], [16]).

For a picocell with a small footprint, there is typically only a small number of nearby PUs. In underlay coexistence, the burden of interference control solely rests upon the secondary network, i.e., the picocells. To identify multiple PUs in the absence of any explicit cooperation, the SUs can exploit the orthogonal pilots transmitted from the PUs or location techniques based on existing spectrum sensing algorithms [17]. We will focus on the case when the SUs are transmitting to the pico BSs while PUs are receiving from the macro BS as this is the most challenging scenario for interference control. It is straightforward to extend the proposed solution to the cases with other uplink and downlink directions.

Since the picocells are independent of each other, we will focus on one picocell and maximize the spectrum efficiency for the SUs inside the picocell while limiting their interference to each nearby PU. Denote N and J as the number of SUs in the picocell and its nearby PUs respectively. Suppose the transmission bandwidth allocated to this picocell is M resource blocks (RBs). We use the weighted sum of channel capacity (over all SUs in the picocell) as the objective function to address both throughput and fairness. We will design a scheduling algorithm that will be run at the beginning of each Transmission Time Interval (TTI) to determine how RBs are to be allocated and the corresponding transmission powers from the SUs. To meet 5G's timing requirements, the running time for the proposed scheduling algorithm must be no greater than one TTI (250 μs).

Based on this understanding, we may drop the notation of time for ease of exposition when there is no ambiguity. Denote a binary variable x_{iB}^m as whether or not SU i will transmit on RB m (under our scheduling algorithm). That is $x_{iB}^m = 1$ if SU i will transmit on RB m and $x_{iB}^m = 0$ otherwise. The "B" in the subscript stands for "Base station". We consider single user OFDMA where each RB can be assigned to at most one SU, i.e.,

$$\sum_{i\in\mathcal{N}} x_{iB}^m \le 1 \qquad (m \in \mathcal{M}) \ , \tag{1}$$

where $\mathcal{M}$ denotes the set $\{1, 2, \cdots, M\}$.

Denote p_{iB}^m as the transmission power of SU i on RB m and denote $P_{iB}^{\max}$ as the maximum transmission power of SU i. Then the constraints for internal power control are given by:

$$0 \le p_{iB}^m \le x_{iB}^m P_{iB}^{\max} \qquad (i \in \mathcal{N},\ m \in \mathcal{M}) \ , \tag{2}$$

$$\sum_{m\in\mathcal{M}} p_{iB}^m \le P_{iB}^{\max} \qquad (i \in \mathcal{N}) \ . \tag{3}$$

where $\mathcal{N}$ is the set $\{1, 2, \cdots, N\}$.

Denote g_{ij}^m as the interference channel gain from SU i to PU j on RB m and denote I_j as the interference threshold for PU j. Under CCP, the interference control (external power control) from the SUs to the PUs can be formulated as chance constraints as follows:

$$\mathbb{P}\left\{\sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}} g_{ij}^m p_{iB}^m \le I_j\right\} \ge 1 - \epsilon_j \qquad (j \in \mathcal{J}) \ , \tag{4}$$

where $\mathcal{J} = \{1, 2, \cdots, J\}$, $\mathbb{P}\{\cdot\}$ denotes probability and ϵ_j is the *risk level* (probability upper bound) for violation of interference threshold I_j. A typical value of ϵ_j ranges from 0.01 to 0.5 depending on the requirement of PU j. In (4), p_{iB}^m's are optimization variables while interference channel gains g_{ij}^m's are modeled as random variables. The set of constraints in (4) states that the aggregate interference from the SUs to PU j over all RBs should stay below threshold I_j with a probability of at least $1 - \epsilon_j$. For generality, we assume only the mean and covariance of g_{ij}^m's are known since these statistics are rather time-invariant compared to instantaneous values of g_{ij}^m's.

For ease of exposition, we rewrite (4) in the matrix form:

$$\mathbb{P}_{\mathbf{g}_j\sim(\overline{\mathbf{g}}_j,\mathbf{R_j})}\left\{\mathbf{g}_j^T\mathbf{p} \le I_j\right\} \ge 1 - \epsilon_j \qquad (j \in \mathcal{J}) \ , \tag{5}$$

where superscript "T" denotes transposition. $\mathbf{p}$ is a $MN \times 1$ column vector consisting of MN transmission powers from the SUs (over all RBs) and is given as

$$\mathbf{p} = \left[p_{1B}^1, \cdots, p_{1B}^M, p_{2B}^1, \cdots, p_{2B}^M, \cdots, p_{NB}^1, \cdots, p_{NB}^M\right]^T \ . \tag{6}$$

$\mathbf{g}_j$ is a $MN \times 1$ random column vector and is defined as

$$\mathbf{g}_j = \left[g_{1j}^1, \cdots, g_{1j}^M, g_{2j}^1, \cdots, g_{2j}^M, \cdots, g_{Nj}^1, \cdots, g_{Nj}^M\right]^T \ , \tag{7}$$

which represents MN random interference channel gains from the SUs (over all RBs) to PU j. $\overline{\mathbf{g}}_j$ (a $MN \times 1$ column vector) and $\mathbf{R}_j$ (a $MN \times MN$ matrix) are the known mean and covariance of $\mathbf{g}_j$ respectively.

By normalizing the bandwidth of each RB to 1 unit, we have the following chance-constrained problem formulation:

$$\begin{aligned}\text{(P1)} \quad & \max_{x_{iB}^m, p_{iB}^m} \ \sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}} w_i \log_2(1 + h_{iB}^m p_{iB}^m) \\ & \text{s.t.} \quad \text{Deterministic Constraints } (1) - (3) \ , \\ & \qquad\ \ \text{Chance Constraints } (5) \ ,\end{aligned}$$

where h_{iB}^m is the transmission channel gain of SU i toward the pico BS on RB m and w_i is the given weight of SU i.

P1 is intractable due to the unknown distribution of $\mathbf{g}_j$ in chance constraints (5). An effective method is to substitute (5) with deterministic constraints. The state-of-the-art technique to perform this substitution is called *Exact Conic Reformulation* (ECR) [10], which replaces the intractable chance constraints (5) with convex deterministic constraints without any relaxations. Comparing to other approaches such as Chebyshev inequality and Bernstein Approximation, ECR requires fewer assumptions and offers better performance. Based on ECR, it can be shown that constraints (5) are mathematically equivalent (w.r.t. $\mathbf{p}$) to the following deterministic constrains [10]:

$$\sqrt{\frac{1-\epsilon_j}{\epsilon_j}}\sqrt{\mathbf{p}^T\mathbf{R}_j\mathbf{p}}+\overline{\mathbf{g}}_j^T\mathbf{p} \le I_j \qquad (j \in \mathcal{J})\ . \tag{8}$$

Replacing (5) with (8) in P1, we have a deterministic maximization problem as the following:

$$\text{(P2)} \quad \max_{x_{iB}^m, p_{iB}^m} \sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}} w_i \log_2(1+h_{iB}^m p_{iB}^m)$$
$$\text{s.t.} \quad \text{Deterministic Constraints } (1)-(3),\ (8)\ .$$

We can use a commercial solver (e.g., Gurobi on CPU) to solve P2 and the computational time is on the order of seconds even for small scale networks (see Section IV). The fundamental issue with commercial solver is the use of sequential iterations, which is time-consuming.

III. A Novel Real-time Solution

In this section, we present our real-time scheduler – GUC, which is designed on off-the-shelf GPU platforms. In Section IV, we will show that GUC offers 10,000 times reduction in computational time (comparing to Gurobi on CPU) while delivering a competitive performance on the objective function.

A. Design Overview

An off-the-shelf GPU typically has thousands of CUDA cores that can be used to run a large number of tasks in parallel. To exploit such parallelism on GPU to solve an optimization problem, the general idea is to decompose the original problem into a large number of small subproblems that can be solved in parallel. If the number of subproblems is too large, we will need to select a subset of these subproblems to solve (with perhaps some loss in performance). After the subset of problems is solved, we can pick the best feasible solution in the hope that it is close to the optimal solution.

Although the above main idea is easy to grasp, how to accomplish each step successfully is far from trivial and constitutes the main efforts of our design.

B. Decomposition of P2

There are two sets of decision variables in P2: the binary variables x_{iB}^m's for RB allocations and the continuous variables p_{iB}^m's for transmission powers. Clearly, x_{iB}^m's are dominant variables and should be considered first.

Our decomposition of P2 is based on finding all feasible RB allocations and correspondingly setting x_{iB}^m's values to 0 or 1. Once x_{iB}^m's are fixed under a feasible RB allocation, we have a subproblem instance of P2 that involves only p_{iB}^m's. By (1) (i.e., RB m can only be allocated to at most one SU), a feasible solution should have no more than one non-zero element in set $\{x_{1B}^m, x_{2B}^m, \cdots, x_{NB}^m\}$. Denote π_m as the SU that is allocated with RB m. Then we have $\pi_m \in \mathcal{N}$ for $m \in \mathcal{M}$; $x_{iB}^m = 1$ if $i = \pi_m$ and $x_{iB}^m = 0$ otherwise. Denote $\pi = \{\pi_1, \pi_2, \cdots, \pi_M\}$ as the indexes of SUs to which the M RBs are assigned. Clearly, for $m \in \mathcal{M}$, $p_{iB}^m > 0$ if $i = \pi_m$ and $p_{iB}^m = 0$ otherwise. Thus, a subproblem for a given π is:

$$\text{(P3)} \quad \max_{p_{\pi_m B}^m} \sum_{m\in\mathcal{M}} w_{\pi_m} \log_2(1+h_{\pi_m B}^m p_{\pi_m B}^m)$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}, \pi_m = i} p_{\pi_m B}^m \le P_{iB}^{\max} \qquad (i \in \pi) \tag{9}$$

$$\sqrt{\frac{1-\epsilon_j}{\epsilon_j}}\sqrt{\mathbf{p}_\pi^T\mathbf{R}_{\pi j}\mathbf{p}_\pi}+\overline{\mathbf{g}_{\pi j}}^T\mathbf{p}_\pi \le I_j \quad (j \in \mathcal{J}) \tag{10}$$

$$p_{\pi_m B}^m \ge 0 \qquad (m \in \mathcal{M}) \tag{11}$$

The $M \times 1$ column vector $\mathbf{p}_\pi = [p_{\pi_1 B}^1, p_{\pi_2 B}^2, \cdots, p_{\pi_m B}^M]^T$ represents the transmission powers corresponding to a given RB allocations. Denote a $M \times 1$ random column vector as $\mathbf{g}_{\pi j} = [g_{\pi_1 j}^1, g_{\pi_2 j}^2, \cdots, g_{\pi_m j}^M]^T$. Then let $\overline{\mathbf{g}_{\pi j}}$ (a $M \times 1$ column vector) and $\mathbf{R}_{\pi j}$ (a $M \times M$ matrix) be its mean and covariance respectively, which are truncated from $\overline{\mathbf{g}}_j$ and $\mathbf{R}_j$. Note that constraints (9) and (10) are modified from constraints (3) and (8) respectively.

C. Select Subproblems

Since each RB can only be allocated to one SU, the total number of possible π is N^M (enumerating all feasible assignments of x_{iB}^m's). Due to the stringent timing requirement, we can only compute a subset of subproblems. Denote K_x as the number of selected subproblems where $K_x \ll N^M$. In the rest of this section, we show how to select these K_x subproblems.

1) Choosing a promising space: The first step is to identify a "promising" space. A promising space can be defined as a subspace (within the original problem space) that contains a set of subproblems with reasonably good (acceptable) performance. It is possible that the optimal solution may fall outside this promising space. But as long as a feasible solution in the promising space is reasonably good (acceptable), it will serve our purpose.

To identify such a promising space, we propose to limit the number of possible RB allocations for each RB from N to a smaller number, denoted as D, i.e., $D < N$. Intuitively, we consider allocating RB m to SU i as a good allocation if w_i and h_{iB}^m are relatively large and g_{ij}^m is relatively small. To exploit this idea, for each x_{iB}^m, we define a metric d_{iB}^m as

$$d_{iB}^m = \frac{w_i h_{iB}^m}{u_{ij}^m} \quad (i \in \mathcal{N},\ m \in \mathcal{M})\ , \tag{12a}$$

$$u_{ij}^m = \sum_{j\in\mathcal{J}} \left\{ \sqrt{\frac{1-\epsilon_j}{\epsilon_j}}\sqrt{||(\mathbf{R}_j)_{((i-1)M+m)*}||_1} + \overline{g_{ij}^m} \right\}\ , \tag{12b}$$

where $(\mathbf{R}_j)_{((i-1)M+m)*}$ is the $((i-1)M+m)$-th column of $\mathbf{R}_j$, $||\cdot||_1$ is the l_1-norm and $\overline{g_{ij}^m}$ is the mean of g_{ij}^m. The intermediate results of $w_i h_{iB}^m$ and d_{iB}^m will be stored in GPU memory for later steps. In (12a), $w_i h_{iB}^m$ is the gradient of the objective function w.r.t. p_{iB}^m at $p_{iB}^m = 0$; u_{ij}^m is related to the interference channel gain g_{ij}^m (with uncertainty), which has a similar form with (8). Taking both into considerations, we use a simple division to define d_{iB}^m.

After calculating the metrics d_{iB}^m for $i = 1, 2, \cdots, N$, we identify the D largest elements from $\{d_{1B}^m, d_{2B}^m, \cdots, d_{NB}^m\}$ as $d_{s_1B}^m \geq d_{s_2B}^m \geq \cdots \geq d_{s_DB}^m$ (in descending order). Denote set $\mathcal{S}_D^m = \{s_1, s_2, \cdots, s_D\}$ and we only consider those $\pi_m \in \mathcal{S}_D^m$ (instead of $\mathcal{N}$ previously). We perform the above procedure for each RB in parallel and obtain its corresponding promising set $\mathcal{S}_D^m$. As each RB only has D possible allocations, we have a promising space $\mathcal{S}$ consisting of D^M subproblems, i.e.,

$$\pi \in \mathcal{S} = \mathcal{S}_D^1 \times \mathcal{S}_D^2 \times \cdots \times \mathcal{S}_D^M \ , \qquad (13)$$

where $\times$ denotes Cartesian product.

A larger value of D expands the promising space while a smaller D shrinks the promising space. We propose to determine a suitable value of D from experiments and in practice, D can be updated based on historical results. After solving P2 in the original space and the promising space under different values of D, we can assess the performance loss in the promising space, which is a function of D. Then we pick the smallest D that offers an acceptable (or near-optimal) performance.

From an implementation perspective, since D is much smaller than N, we can perform D times of parallel reduction to obtain $\mathcal{S}_D^m$. Parallel reduction is widely used in GPU programming [18] to reduce computation time especially for operations such as finding the maximum (or minimum) or the sum of some given numbers. In GUC, we will employ parallel reduction whenever possible to reduce computation time.

2) Sampling: If $K_x < D^M$, we need to perform sampling to obtain K_x subproblems. Otherwise, we can skip this step. There are D^M events in $\mathcal{S}$ and each event corresponds to one subproblem instance. For an event $\varphi = \{\varphi_1, \varphi_2, \cdots, \varphi_M\} \in \mathcal{S}$, we define $d_\varphi = d_{\varphi_1 B}^1 \cdot d_{\varphi_2 B}^2 \cdots d_{\varphi_M B}^M$. Then we assign event φ with the following probability:

$$\mathbb{P}\{\varphi\} = \frac{d_\varphi}{\sum\limits_{\phi \in S} d_\phi} \qquad (\varphi \in \mathcal{S}) \ . \qquad (14)$$

We then obtain K_x samples following the probabilities given in (14) and obtain K_x events (subproblems).

D. Solving Subproblems

Each subproblem (in the form of P3) is a convex optimization problem with M continuous decision variables $p_{\pi_m B}^m$'s. A conventional solution approach to solve such a convex problem is to use gradient-based iterations. However, such an approach usually requires tens of iterations to converge and cannot meet the 250 μs timing requirement.

We propose to solve P3 based on decent starting points and a simple scaling-based local search. Suppose for each subproblem, we generate K_p starting points (each with M transmission powers) given by

$$\begin{aligned}
\mathbf{p}_\pi^k &= [p_{\pi_1 B}^{1k}, p_{\pi_2 B}^{2k}, \cdots, p_{\pi_M B}^{Mk}]^T \ (k = 1, 2, \cdots, K_p) \ , && (15a)\\
p_{\pi_m B}^{mk} &= \frac{w_{\pi_m} h_{\pi_m B}^m}{\sum\limits_{m \in \mathcal{M}} w_{\pi_m} h_{\pi_m B}^m} \cdot \frac{\sum\limits_{j \in \mathcal{J}} I_j}{\sum\limits_{j \in \mathcal{J}} u_{\pi_m j}^m} + \delta v_k^m && (15b)\\
&= \frac{d_{\pi_m B}^m \sum\limits_{j \in \mathcal{J}} I_j}{\sum\limits_{m \in \mathcal{M}} w_{\pi_m} h_{\pi_m B}^m} + \delta v_k^m \qquad (m \in \mathcal{M}) \ , && (15c)
\end{aligned}$$

where v_k^m is a uniformly distributed random variable and δ is a small positive number that ensures all K_p starting points are within a sphere. (15c) is easy to calculate since the results of $w_{\pi_m} h_{\pi_m B}^m$ and $d_{\pi_m B}^m$ are already stored in GPU memory per our earlier discussion.

However, the starting points in (15) guarantee neither feasibility nor performance. Thus, we propose a scaling-based local search to refine each starting point. For simplicity, we drop the notation of k in the rest of this section. This local search contains three scaling procedures, as detailed below.

1) First scaling: The i-th constraint in (9) states that the total transmission power from SU i cannot exceed its device limit. Given the transmission powers from a starting point, we calculate the current total transmission power from SU i as

$$P_{iB}^{\text{total}} = \sum_{m \in \mathcal{M}, \pi_m = i} p_{\pi_m B}^m \quad (i \in \pi) \ . \qquad (16)$$

We will scale down the transmission powers from SU i only if $P_{iB}^{\text{total}} > P_{iB}^{\max}$. Thus, the first scaling can be summarized as

$$p_{\pi_m B}^m = p_{\pi_m B}^m \cdot \min\left\{1, \frac{P_{\pi_m B}^{\max}}{P_{\pi_m B}^{\text{total}}}\right\} \qquad (m \in \mathcal{M}) \ . \qquad (17)$$

After (17), constraints (9) are guaranteed to be feasible since no SU will exceed its device limit of total transmission power. Denote $\pi' = \{i : i \in \pi, P_{iB}^{\max} \leq P_{iB}^{\text{total}}\}$ as the set of SUs who have already reached their device limit, which will be used in the second scaling.

2) Second scaling: To maximize spectrum efficiency, usually one of the constraints in (10) should be binding, meaning the SUs have taken full advantage of the transmission opportunities allowed by PUs. Thus, in the second scaling, in addition to ensuring the feasibility of constraints (10), we will try to improve the objective value by scaling up the transmission powers if possible.

We first calculate the current interference level from the SUs in π (denoted as $\hat{I}_j$) and the average interference from the SUs in π' (denoted as $\hat{I}'_j$) as follows:

$$\begin{aligned}
\hat{I}_j &= \sqrt{\frac{1-\epsilon_j}{\epsilon_j}} \sqrt{\mathbf{p}_\pi^T \mathbf{R}_{\pi j} \mathbf{p}_\pi} + \overline{\mathbf{g}_{\pi j}}^T \mathbf{p}_\pi \ , \\
\hat{I}'_j &= \sum_{m \in \mathcal{M}, \pi_m \in \pi'} \overline{g_{\pi_m j}^m} p_{\pi_m B}^m \ ,
\end{aligned} \qquad (18)$$

where $\overline{g_{\pi_m j}^m}$ is the mean of $g_{\pi_m j}^m$ (m-th element of $\overline{\mathbf{g}_{\pi_m j}^m}$). We can reduce the computation time of (18) by exploiting the symmetric and sparse properties of $\mathbf{R}_{\pi j}$. For instance, $\mathbf{R}_{\pi j}$ is a diagonal matrix when the interference channel gains $g^m_{\pi_m j}$'s are independent of each other. In correlated channels, $\mathbf{R}_{\pi j}$ can be characterized as a band matrix since the correlation between two RBs decreases as they are further apart.

There are two cases when comparing current interference level $\hat{I}_j$ with the interference threshold I_j.

Case 1: $I_j \leq \hat{I}_j$. This means that j-th constraint in (10) is not satisfied. We need to scale down all $p^m_{\pi_m B}$'s by $I_j/\hat{I}_j$.

Case 2: $I_j > \hat{I}_j$. In this case, we can further increase the transmission powers from the SUs. For those SUs in set π', we cannot increase their transmission powers since they have already reached their device limits. Thus, we will scale up transmission powers from the SUs in $\pi\backslash\pi'$ by $(I_j - \hat{I}'_j)/(\hat{I}_j - \hat{I}'_j)$.

Specifically, denote λ_j as the scaling factor obtained from the j-th constraint of (10) and it is calculated as

$$\lambda_j = \begin{cases} \dfrac{I_j}{\hat{I}_j} & \text{if } I_j \leq \hat{I}_j \ , \\ \dfrac{I_j - \hat{I}'_j}{\hat{I}_j - \hat{I}'_j} & \text{if } I_j > \hat{I}_j \ . \end{cases} \tag{19}$$

We perform the second scaling for RB m as

$$p^m_{\pi_m B} = \begin{cases} p^m_{\pi_m B} \cdot \min\{1, \lambda_1, \cdots, \lambda_J\} & \text{if } \pi_m \in \pi' \ , \\ p^m_{\pi_m B} \cdot \min\{\lambda_1, \cdots, \lambda_J\} & \text{otherwise} \ . \end{cases} \tag{20}$$

After the second scaling based on (20), constraints (10) are feasible but constraints (9) may not be satisfied. The proof of this claim is omitted due to space limitation.

3) Third scaling: To ensure constraints (9) holds, we will repeat the first scaling given by (16) and (17). The obtained transmission powers will be a feasible solution for the current subproblem. A key observation is that the third scaling will not increase transmission powers $p^m_{\pi_m B}$'s and hence constraints (10) remain feasible. Further, constraints (11) only state the non-negativity of $p^m_{\pi_m B}$'s, which is trivially true. Thus, we have reached a feasible solution from a starting point for the current subproblem and it is a feasible solution for P2 and P1.

By doing the above procedure for all K_x subproblems (each with K_p starting points) in parallel, we have $K_x K_p$ feasible solutions for P2 (and P1) at hand. The feasible solution with the highest objective value will be the final solution from GUC.

E. Moving Data between CPU and GPU

Before the decomposition step, we need to transfer data (parameters) from the CPU memory to GPU memory. After we find the best solution, we also need to transfer this solution from the GPU memory back to CPU memory. Since we are using off-the-shelf GPU, such data transfer will go through PCIe interface between CPU and GPU. To reduce this data transfer time, asynchronous data transfer is employed between CPU and GPU [19].

IV. Implementation and Experimental Results

In this section, we implement our proposed GUC and conduct experiments to validate its performance. Our performance evaluation focuses on elapsed time and objective value (achievable spectrum efficiency).

A. Implementation Platforms

We implement GUC on NVIDIA Quadro P6000 which has 3840 CUDA cores. Due to space limitation, we omit the details of our implementation efforts regarding threads allocation, memory management, threads synchronization and kernel functions, etc.

For comparison to CPU platform, we use 16-core Intel Xeon E5-2687w and obtain a tight upper bound [20] of P2 using Gurobi 8.0.1, which serves as a benchmark in this paper.

B. Experiment Settings

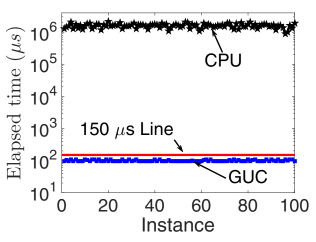
We assume the following input data to GUC are available: network topology, transmission power limit, thermal noise and interference threshold. Specifically, the distance between the macro BS and the pico BS is 400 meters and the radius of the picocell is 50 meters. The transmission power of macro BS on each RB is 46 dBm and each SU has a maximum transmission power of 20 dBm across all RBs. The thermal noise on each RB is 1×10^{-7} mW. The pico BS is located at the origin. There are five PUs ($J = 5$) near the picocell and the coordinates of the PUs are $(60, 0), (-50, 25)$, $(-40, -40), (10, 55)$ and $(35, -45)$ (all in meters). For simplicity, we use the same interference threshold $I_j = 3\times10^{-7}$ mW for all PUs. The path loss for macro- and picocells are based on ITU outdoor and indoor path loss model respectively [4] and we use Rayleigh fading as the fast fading.

C. Results

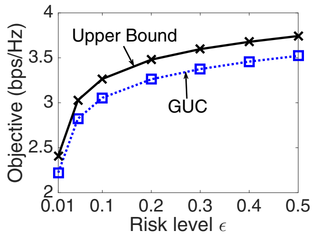
The results shown in this section are the average of 500 runs. For each run, we randomly generate the weights w_i's and the network topology based on uniform distributions. The weights w_i's are normalized over all SUs so that the average weight of each SU is 0.1. Based on the above channel model, we generate one sample of transmission channel gains h^m_{iB}'s and 10,000 samples of the interference channel gains g^m_{ij}'s. The 10,000 samples of g^m_{ij}'s are used to calculate the mean $\overline{\mathbf{g}}_j$ and covariance $\mathbf{R}_j$. Although GUC also works under correlated channels, due to space limitation, we only show results when g^m_{ij}'s are independent with each other.

1) Moderate Size: We consider a moderate size picocell with 10 SUs and 16 RBs (i.e., $N = 10, M = 16$). For the design parameter D, we set $D = 2$ as it can maintain 99% optimality on average. We choose 32 subproblems for each run ($K_x = 32$) and use 64 different starting points ($K_p = 64$) for each subproblem with $\delta = 0.2$.

Fig. 1 shows our experimental results. A snapshot of 100 runs is shown in Fig. 1(a). The computational time of GUC is 99.5 μs on average (with 6.0 μs standard deviation). The maximum computational time of GUC is no more than 150 μs. In contrast, the computational time on CPU is $\sim 10^6$ μs,

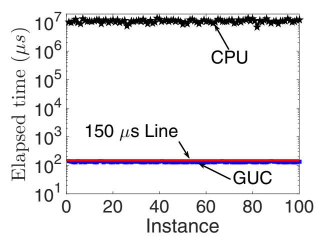


(a) Computational time

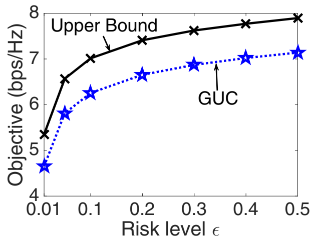


(b) Spectrum efficiency

Fig. 1: GUC's performance when $N = 10$ and $M = 16$.



(a) Computational time



(b) Spectrum efficiency

Fig. 2: GUC's performance when $N = 20$ and $M = 32$.

which is about $10,000$ longer than that of GUC. Fig. 1(b) shows GUC's performance in achievable objective values. It shows that GUC can achieve >90% optimality on average. (with 2.7% standard deviation). We also verify that the actual threshold violation probability of each PU is indeed smaller than the risk level.

2) Large Size: We also test a large size picocell with $N = 20$ and $M = 32$. Under the setting of a picocell, such scale can represent the extreme case. The design parameters are the same with that in moderate size setting, i.e., $D = 2$, $K_x = 32$, $K_p = 64$, $\delta = 0.2$. The results are shown in Fig. 2.

Fig. 2(a) is a snapshot of 100 runs. The computational time of GUC is 131.6 μs on average (with 4.3 μs standard deviation). All computational times of GUC are smaller than 150 μs while that on CPU is $\sim 10^7$ μs, which is over $10,000$ times greater than that of GUC. The achievable spectrum efficiency shown in Fig. 2(b) is 89.5% within the upper bound on average (with 3.5% standard deviation).

V. Conclusions

In this paper, we studied underlay coexistence and employed CCP to address channel gain uncertainty. To ensure that the computational time of a solution can meet 5G's real-time requirement, we designed and implemented GUC on GPU platforms. By solving subproblems in parallel on a large number of GPU cores, GUC successfully achieved an overall computation time under $150\mu s$ – a 10,000 times reduction in computation time. Further, the objective values achieved by GUC is 90% of the upper bound of optimal objectives on average. Our design is the first practical solution to solve the CCP problem for underlay coexistence in real-time.

VI. Acknowledgments

This research was supported in part by U.S. Army Research Laboratory under Cooperative Agreement No. W911NF1820293. The work of Y.T. Hou and W. Lou was also supported in part by NSF under grants 1642873 and 1800650.

References

[1] A. Goldsmith, S. A. Jafar, I. Maric, and S. Srinivasa, "Breaking spectrum gridlock with cognitive radios: An information theoretic perspective," *Proceedings of the IEEE*, vol. 97, no. 5, pp. 894–914, 2009.

[2] D. J. Costello and G. D. Forney, "Channel coding: The road to channel capacity," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1150–1177, 2007.

[3] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: A review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, 1998.

[4] 3GPP, *3GPP TR 36.931: Radio Frequency (RF) requirements for LTE Pico Node B*, June 2018, version 15.0.0. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2589.

[5] ——, *3GPP TR 36.932: Scenarios and requirements for small cell enhancements for E-UTRA and E-UTRAN*, July 2018, version 15.0.0. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2590.

[6] W. W. Li, Y. J. Zhang, A. M.-C. So, and M. Z. Win, "Slow adaptive OFDMA systems through chance constrained programming," *IEEE Trans. Signal Processing*, vol. 58, no. 7, pp. 3858–3869, 2010.

[7] S. Wang, W. Shi, and C. Wang, "Energy-efficient resource management in OFDM-based cognitive radio networks under channel uncertainty," *IEEE Trans. Communications*, vol. 63, no. 9, pp. 3092–3102, 2015.

[8] N. Y. Soltani, S.-J. Kim, and G. B. Giannakis, "Chance-constrained optimization of OFDMA cognitive radio uplinks," *IEEE Trans. Wireless Communications*, vol. 12, no. 3, pp. 1098–1107, 2013.

[9] N. Mokari, S. Parsaeefard, P. Azmi, H. Saeedi, and E. Hossain, "Robust ergodic uplink resource allocation in underlay OFDMA cognitive radio networks," *IEEE Trans. Mobile Computing*, vol. 15, no. 2, pp. 419–431, 2016.

[10] S. Li, Y. Huang, C. Li, B. A. Jalaian, Y. T. Hou, and W. Lou, "Coping uncertainty in coexistence via exploitation of interference threshold violation," in *Proc. ACM MobiHoc 2019*, Catania, Italy, July 2019.

[11] 3GPP, *3GPP TR 38.211: Physical channels and modulation*, March 2019, version 15.5.0. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3213.

[12] Qualcomm. The 3GPP release-15 5G NR design. Available: https://www.qualcomm.com/media/documents/files/the-3gpp-release-15-5g-nr-design.pdf (Last accessed: Aug., 2019).

[13] Y. Huang, S. Li, Y. T. Hou, and W. Lou, "GPF: A GPU-based design to achieve ~100 μs scheduling for 5G NR," in *Proc. MobiCom 2018*, New Delhi, India, July 2019.

[14] A. Damnjanovic, J. Montojo, Y. Wei, T. Ji, T. Luo, M. Vajapeyam, T. Yoo, O. Song, and D. Malladi, "A survey on 3GPP heterogeneous networks," *IEEE Wireless Communications*, vol. 18, no. 3, pp. 10–21, 2011.

[15] F. Jin, R. Zhang, and L. Hanzo, "Fractional frequency reuse aided twin-layer femtocell networks: Analysis, design and optimization," *IEEE Trans. Communications*, vol. 61, no. 5, pp. 2074–2085, 2013.

[16] H.-B. Chang and I. Rubin, "Optimal downlink and uplink fractional frequency reuse in cellular wireless networks," *IEEE Trans. Vehicular Technology*, vol. 65, no. 4, pp. 2295–2308, 2016.

[17] A. Furtado, L. Irio, R. Oliveira, L. Bernardo, and R. Dinis, "Spectrum sensing performance in cognitive radio networks with multiple primary users," *IEEE Trans. Vehicular Technology*, vol. 65, no. 3, pp. 1564–1574, 2016.

[18] M. Harris. Optimizing parallel reduction in CUDA. Available: https://developer.download.nvidia.com/assets/cuda/files/reduction.pdf (Last accessed: Aug., 2019).

[19] ——. How to overlap data transfers in CUDA C/C++. Available: https://devblogs.nvidia.com/how-overlap-data-transfers-cuda-cc/ (Last accessed: Aug., 2019).

[20] Y. T. Hou, Y. Shi, and H. D. Sherali, *Applied Optimization Methods for Wireless Networks*, Chp. 5, pp. 110-112. Cambridge University Press, 2014.