Using Dynamic Programming to Optimize Cellular Networks Modeled as Graphical Games

Artur Popławski, and Szymon Szott

***Abstract*—Cellular networks are often modeled using game theory, with base stations as players contending for a shared resource (the radio channel). Alternatively, if base stations are considered as nodes joined by edges (which represent significant interference), we obtain a graph structure. A game represented in this way is called a graphical game. We explore this representation by decomposing the network graph through tree decomposition and apply dynamic programming to find the optimum welfare, i.e., a resource allocation strategy profile most effective from the point of view of the overall network performance. We verify our approach through simulations and discuss the possibility of implementing this solution in a distributed manner.**

***Index Terms*—cellular networks, game theory, graphical games, LTE, optimization, welfare, tree decomposition**

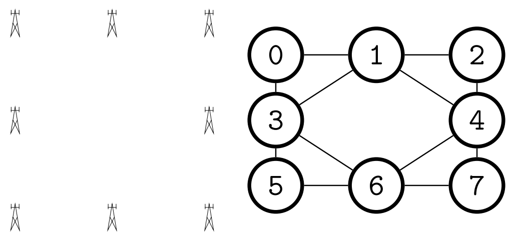

Fig. 1. An arrangement of wireless base stations (left) and their corresponding graph (right). The edges connect nodes corresponding to base stations whose distance from each other does not exceed, in this case, half of the length of the diagonal between the stations furthest apart from each other.

I. Introduction

OPTIMISING the performance of a single base station (BS) in a cellular network is a relatively well understood problem. In an OFDM-based cellular network, such as Long Term Evolution (LTE) or New Radio (NR), the core part of this optimization task is to find the allocation of time and frequency resources to optimally fulfill throughput demands considering current radio conditions, fairness between user equipment (UEs), etc. Optimising the global performance of a network is a completely different problem. A basic issue is that transmitters interact, i.e., a single BS receives both the signal dedicated to it and signals from other BSs. On a rudimentary level, the activity of these other transmitters adds to the noise in the channel formed between the transmitter and receiver and as such is destructive to the transmission. Advanced mechanisms such as coordinated multi-point (CoMP) [1] or distributed massive multiple-input and multiple-output (MIMO) [2] are designed to gain from these interactions. In this paper, however, we focus on network performance optimisation by minimising the negative effects of interference rather than gaining from them. We also work under the approximation that only the most interfering transmitters (typically the closest ones) are considered. Thus, the network is considered as a graph where nodes represent BSs and edges connect the most severely interfering ones (Fig. 1).

A situation where there is an internal conflict between agents engaged in a joint activity is naturally modeled using game theory [3], [4]. We consider a sequence of decisions made by a BS, which can be an evolved Node B (eNB) in the LTE case or a next generation Node B (gNB) in the NR case. Each decision is of the form “transmit” or “do not transmit” at a given moment. The “do not transmit” decision in this framework does not arise from the fact that there is no data to be sent. It is rather a kind of “sacrifice” made by the BS to reduce interference and increase the overall performance of the network. The strategy applied by the BS is a probability distribution over these two possible decisions. In other words, at any time the decision at the transmitter is made randomly with a certain probability and this probability is considered its strategy. Since decisions made by different transmitters are independent, this brings us into the realm of games in mixed strategies. Thus, the optimisation domain in our model is the set of possible assignments of mixed strategies to all BSs and the utility function of interest is the average throughput of the whole network.

To find the optimal network operating point, we propose an algorithm within the dynamic programming paradigm. Dynamic programming is now a commonly used tool in algorithm design, network science, control theory, and others [5]. The specific method presented here is an adaptation of an algorithm for computing a mixed strategy Nash equilibrium in a graphical game where the graph structure associated with the game is a tree [6]. We apply this algorithm to the problem of finding the optimal assignment of a transmission strategy for each BS. We relax the assumption of [6] about the graph structure, as we consider general graphs, not only trees. To consider general graphs, we turn our attention to the *tree decomposition of graphs*, which constructs trees that approximate arbitrary graphs. Another inspiration to use tree decomposition is the dynamic programming algorithms of [6], [7] and the methods described in [8].

A. Popławski is with Nokia Technology Center Krakow, Poland and AGH University of Science and Technology, Kraków, Poland (E-mail: artur.poplawski@nokia.com)

S. Szott is with AGH University of Science and Technology, Kraków, Poland (E-mail: szott@agh.edu.pl)

This work was partially supported by the Polish Ministry of Science and Higher Education with the subvention funds of the Faculty of Computer Science, Electronics, and Telecommunications of AGH University.

DOI: 10.36244/ICJ.2022.4.9

The tree decomposition, as well as the associated notion of *treewidth* (the minimum width of a tree decomposition, cf. Section II-D), has been studied extensively since its introduction [9]. We refer the reader to [10] for a graph-theoretic perspective on tree decomposition and treewidth. From an algorithmic perspective, having a precomputed tree decomposition of a graph where the treewidth of the decomposition is small, dynamic programming allows computing functions of a graph where the computation time depends on the size of the graph as a small-degree polynomial. This approach reduces the problem of efficiently calculating a function of the graph to finding a good tree decomposition of the graph. Obviously, not all graphs have a good decomposition and in general it is not easy to find such a decomposition even if they have one. However, algorithms have been proposed that allow us to find the decomposition quickly if the treewidth is small: [11]–[13].

Methods based on dynamic programming for good tree decomposition, in addition to having theoretical significance, have found their way to practical engineering and computing. An early review of applications, including those beyond theoretical interest, is given in [14]. In the area strictly related to telecommunications, it is worth mentioning [15] in which applications have been developed to the problem of routing in wireless sensor networks.

We provide a dynamic programming algorithm that finds the optimal assignment of strategies, from the point of view of global performance, given its tree decomposition. As an immediate application of the algorithm, motivating this study, we propose a mechanism of downlink transmission optimisation in an LTE or NR cellular network by changing the activity of BSs in time and reducing the negative impact of interference on network performance.

The remainder of the work is structured as follows. In Section II, we recall basic definitions related to combinatorics and game theory, including the notion of tree decomposition. Then, in Section III, the main algorithm is proposed for an abstract setting. Next, we describe the realization of the algorithm in a network of interactive devices considering distributed optimisation (Section IV). We also discuss the application of the method to solve the problem of channel allocation in cellular networks (Section V), which constitutes the main motivation for this work, and validate our approach with simulations (Section VI). Finally, Section VII summarises the results and gives insight into the perspective of further research and applications.

II. Formal Definitions and Notation

In this section, we present notational conventions used later in the text and basic definitions related to graphs, game theory, and tree decomposition, which are relevant to the modelling and analysis of cellular networks.

A. Conventions

If P is a set, then by $\{S_p\}_{p\in P}$ we understand the family of sets indexed by P. Having such a family, by $\prod_{p\in P} S_p$ we understand the Cartesian product indexed by P, i.e., $\prod_{p\in P} S_p = \{f : P \to \bigcup_{p\in P} S_p, \text{s.t. for each } p,\ f(p) \in S_p\}$.

Having $K \subset P$ and the family $\{S_p\}_{p\in P}$, by $\pi_K : \prod_{p\in P} S_p \to \prod_{k\in K} S_k$ we understand the projection operator, i.e., for $f \in \prod_{p\in P} S_p$ and $x \in K$, $\pi_K(f)(x) = f(x)$. When using the Cartesian product, we label factors of the product with an index taken from a set. The set-theoretic operation of disjoint sum on the indexing sets easily transfers to the product. Namely, if $K \cap L = \emptyset$, $(K \cup L) \subset P$, we have $\prod_{k\in K} S_k \times \prod_{l\in L} S_l = \prod_{k\in K\cup L} S_k$. We also disregard the order when referring to the elements of such a product, i.e., in our convention with the same assumptions about sets K and L as above we have: if $x \in \prod_{k\in K} S_k$, $y \in \prod_{l\in L} S_l$ then we denote $(x, y) = (y, x) = z \in \prod_{k\in K} S_k \times \prod_{l\in L} S_l = \prod_{l\in K\cup L} S_l$. To simplify exposition of the algorithm, we formally allow the Cartesian product to be taken over empty sets. By convention, a product over an empty set is the neutral element for the operation on Cartesian products: $\prod_{k\in K} S_k \times \prod_{l\in\emptyset} S_l = \prod_{k\in K} S_k$. With this convention, if formally $s' \in \prod_{l\in\emptyset} S_l$, we have $(s, s') = s$. For the $\arg\max$ operator we take the convention that for the function $f : \prod_{k\in K} S_k \to \mathbb{R}$, $s \in \prod_{k\in K} S_k$ we define $\arg\max_{s'\in\prod_{l\in\emptyset} S_l} f(s', s) = f(s)$. For function $f : X \to Y$, we refer to X also by $dom(f)$.

B. Graphs

Definition 1. *An* undirected graph G *is a pair* $G = (V, E)$ *where* V *is a finite set of vertices (representing BSs),* E *is the set of edges (representing interference), and* $E \subset \{e \subset V : |e| = 2\}$.

We consider only undirected graphs, which we later refer to simply as *graphs*.

Definition 2. *A* path *between elements* $x, y \in V$ *in graph* (E, V) *is a sequence* $x = v_0, \dots, v_{k-1} = y$*, where for each* $0 < i \le j < k$ *we have:*

1) *if* $i \ne j$ *then* $v_i \ne v_j$,
2) $\{v_{i-1}, v_i\} \in E$.

Having graph $G = (V, E)$ and vertex $v \in V$ we have a function $nb_G : 2^V \to 2^V$ defined as $nb_G(S) = \{w \in V : \exists v \in S, \{v, w\} \in E\}$. We call $nb_G(S)$ the *neighbourhood* of S in G. For the singleton $\{v\}$, we slightly abuse the notation and simply write $nb_G(v)$ instead of $nb_G(\{v\})$. For convenience, we define the *extended neighbourhood* of S in G by $xnb_G(S) = S \cup nb_G(S)$, preserving the same notation convention for singletons as in the case of the operator nb_G.

Definition 3. *An undirected graph is a tree when there is only one path between any pair of vertices. Let* $G = (V, E)$ *be a tree and consider any arbitrary vertex* $t \in V$*. Then, the pair* (G, t) *is called a* rooted tree *and* t *is called a* root.

For a rooted tree (G, t), we define a function

$$parent : V \setminus \{t\} \to V$$

such that $parent(x) = y$ iff there is a $x, y, \ldots, t$ path in G. We also define a function $children : V :\to 2^V$ such that

$$children(x) = \{y : parent(y) = x\}.$$

We refer to $v \in V$ such that $children(v) = \emptyset$ as *leaves*.

For convenience, we also define a function $offspring : V \to 2^V$ inductively (by induction starting from the leaves) as

$$offspring(x) = \emptyset,$$

where x is a leaf, and

$$offspring(x) = children(x) \cup \bigcup_{y \in children(x)} offspring(y).$$

C. Game Theory

Definition 4. *An $n-$player game Γ is a triple*

$$\Gamma = (P, \{S_p\}_{p \in P}, \{u_p\}_{p \in P}),$$

where P is the set of players ($|P| = n$), for each $p \in P$, S_p is the set of strategies available to the players, and $u_p : \prod_{p \in P} S_p \to \mathbb{R}$ is the payoff function.

For a finite set K, we define a simplex $\Delta_K = \{x \in \mathbb{R}^{|K|} : x \geq 0 \wedge \sum_{k \in K} x_k = 1\}$. Elements of the simplex represent possible probability distributions on the set K treated as a finite probability space.

Definition 5. *For game $\Gamma = (P, \{S_i\}_{i \in P}, \{u_i\}_{i \in P})$, where P and each S_p is finite, we call the game Γ* in mixed strategies *and denote by $M(\Gamma)$ the following game:*

$$M(\Gamma) = (P, \{\Delta_{S_i}\}_{i \in P}, \{u_{\Delta,i}\}_{i \in P}),$$

where

$$u_{\Delta,i} : \prod_{i \in P} \Delta_{S_i} :\to \mathbb{R}$$

is defined by

$$u_{\Delta,i}(x) = \sum_{s \in \prod_{j \in P} S_p} \left(\prod_{j \in P} x_j(s_j) \right) u_i(s).$$

To capture the quantitative *influence* of nodes in a game Γ, we define the function $Infl_\Gamma : P \times 2^P \to \mathbb{R}$ as

$$Infl_\Gamma(i, I) = \max_{s_{-I} \in S_{-I}} (\max_{s_I \in S_I} u_i(s_I, s_{-I}) - \min_{s_I \in S_I} u_i(s_I, s_{-I})).$$

We state that $i \in P$ is $\epsilon-$independent from $I \subset P$ if $Infl_\Gamma(i, I) \leq \epsilon$.

The intuitive meaning of this function is as follows: for a given game Γ, player i, and set of players I, it provides information about the maximal possible change of payoff of player i caused by the actions of players from I. If the value of $Infl_\Gamma(i, I)$ is small it means that in practice player i is not affected by other players in I. If the players are wireless BSs and their interaction is the radio interference, this function measures to what extent BSs from the set I interfere with the transmissions from BS i to UEs.

We assume a fixed ϵ and that for each p we have a (non-unique and possibly empty) set of non-influencing players $NI_p \subset P$ such that $p \notin NI_p$ and $Infl_\Gamma(p, NI(p)) \leq \epsilon$. This set contains those elements of P different than p itself, that collectively have little influence on the payoff p despite the action they take. Then, fixing the choice of NI_p for each p, we associate with Γ a graph $(P, E_{\Gamma,\epsilon})$ in the following manner:

$$\{v, w\} \in E_{\Gamma,\epsilon} \Leftrightarrow v \notin NI_w \vee w \notin NI_v.$$

The graph $(P, E_{\Gamma,\epsilon})$ under a suitable choice of NI is called the *ϵ graphical representation of* Γ. For $\epsilon = 0$ it is the *graphical representation of* Γ.

We emphasize once more that an ϵ graphical representation depends not only on the choice of ϵ but also on the choice of NI which is not canonical.

For a game with graphical representation $G = (P, E)$ and for player p, we formally consider u_p not as a function with the domain $\prod_{v \in P} S_v$ but as a function with the domain $\prod_{v \in xnb_G(p)} S_v$. We use the notation $dom_G(p)$ for set $xnb_G(p)$ omitting the subscript G when it is clear from the context. For the ϵ graphical representation G of Γ, there is a graphical game that approximates Γ.

The following theory gives the strongest results for games for which there exist sparse ϵ graphical representations of Γ for small ϵ. Games arising from the modeling of cellular networks where the interaction between the players representing BSs is caused by interference typically satisfy this condition. For any given BS there is a relatively small number (compared to the number of all BSs) of transmitters within the distance where interference influences signal reception.

D. Tree Decomposition

An example of how to apply dynamic programming methods in graphical games where the underlying graph is a tree is given in [6]. To apply algorithms suited to this regular structure in more general settings, we need to transform a general graph structure into a tree. A known technique of representing a general graph as a tree is tree decomposition.

Definition 6. *Let $G = (V, E)$, be a graph. The tree decomposition of G is a graph (X, F) such that:*

1) *(X, F) is a tree,*
2) *$x \subset V$ for each $x \in X$,*
3) *$\bigcup_{x \in X} x = V$,*
4) *for each $e \in E$ there is $x \in X$ such that $e \subset X$,*
5) *for each $v \in V$ if there are $x, y \in X$ such that $v \in x$ and $v \in y$, then for each z in the path from x to y in graph (X, F) we have $v \in z$.*

Having a tree decomposition $T = (X, F)$ we call $\max_{x \in X} |x| - 1$ the *treewidth* of T. The set of all tree decompositions of $G = (V, E)$ is denoted by $TD(G)$. We make the following observation:

Remark 1. *$TD(G) \neq \emptyset$ for any graph G.* □

This is the immediate consequence of the fact that the trivial graph $(\{V\}, \emptyset)$ is always a tree decomposition of $G = (V, E)$.

One can compare Fig. 2b depicting the graph with Fig. 2c depicting one of its tree decompositions to gain an intuitive understanding of the concept.

III. Welfare in Graphical Games

Using the provided definitions, we first provide a scheme for computing the welfare for a graphical game with tree decomposition and then proceed to analyze its performance. The scheme for calculating maximum welfare depends on:

- a graph $G = (P, E)$ describing the structure of the graphical game Γ,
- a particular representative $T \in TD(G)$, i.e., a tree decomposition of G.

Consider $(X, F) = T \in TD(G)$ and a distinguished element $t \in X$ as the root. We start with the following lemma:

Lemma 1. *For the tree $TD(G) \ni T = (X, F)$ rooted at $t \in X$, the following regularity condition is satisfied: for each $x \neq t$, if $v \in x \setminus parent(x)$ and $\{v, w\} \in E$ then*

$$w \in x \cup \bigcup_{z \in offspring(x)} z.$$

Proof. Assume, contrary to the thesis, that there is an $x \in X$ such that there is a $v \in (x \setminus parent(x))$ and $w \in V$ and such that $e = \{v, w\} \in E$ and $w \notin x \cup \bigcup_{z \in offspring(x)} z$. From Definition 6, we know there is a node $y \in X$ such that $e \subset y$. From the assumption we have $y \notin \{x\} \cup offspring(x)$. This means, again from Definition 6, that v belongs to every node of the tree T on the path from y to x, but such a path must contain $parent(x)$. This is a contradiction as we assumed that $v \in x \setminus parent(x)$. □

Consider the following scheme of computing welfare of Γ with operators dependent on the structure of T:

$$FD(x) = xnb_G(parent(x)) \cap \bigcup_{y \in \{x\} \cup offspring(x)} y$$

and

$$BD(x) = x \cup \bigcup_{y \in children(x)} FD(y).$$

Remark 2. *For root t, $FD(t) = \emptyset$.*

Remark 3. $BD(x) = x \cup (xnb_G(x) \cap \bigcup_{y \in offspring(x)} y)$.

Proof.

$$\begin{aligned} BD(x) = x \cup \bigcup_{y \in children(x)} FD(y) = \\ x \cup \bigcup_{y \in children(x)} (xnb_G(x) \cap \bigcup_{z \in \{y\} \cup offspring(y)} z) = \\ x \cup (xnb_G(x) \cap \bigcup_{y \in children(x)} \bigcup_{z \in \{y\} \cup offspring(y)} z) = \\ x \cup (xnb_G(x) \cap \bigcup_{y \in offspring(x)} y \end{aligned}$$

The scheme works in two passes, each with two stages. In the first pass in stage one:

For $x \in X \setminus \{t\}$, let $y = parent(x)$ compute the function:

$$f_{x \to y} : \prod_{v \in FD(x)} S_v \to \mathbb{R}$$

as:

$$\begin{aligned} f_{x \to y}(s) = \\ \max_{s' \in (\prod_{w \in BD(x) \setminus FD(x)} S_w)} \Bigg(\sum_{v \in x \setminus y} u_v(\pi_{xnb_G(v)}(s', s)) + \\ \sum_{z \in children(x)} f_{z \to x}(\pi_{FD(z)}(s', s)) \Bigg) \end{aligned}$$

and the (possibly empty) function

$$m_{x \to y} \quad : \quad \prod_{v \in FD(x)} S_v \quad \to \quad \prod_{w \in BD(x) \setminus FD(x)} S_w$$

as:

$$\begin{aligned} m_{x \to y}(s) = \\ \arg\max_{s' \in \left(\prod_{w \in BD(x) \setminus FD(x)} S_w \right)} \Bigg(\sum_{v \in x \setminus y} u_v(\pi_{xnb_G(v)}(s', s)) + \\ \sum_{z \in children(x)} f_{z \to x}(\pi_{FD(z)}(s', s)) \Bigg). \end{aligned}$$

This ends stage one. At stage two, for root t we compute:

$$\begin{aligned} \max_{s' \in (\prod_{w \in BD(t)} S_w)} \Bigg(\sum_{v \in x \setminus y} u_v(\pi_{xnb_G(v)}(s', s)) + \\ \sum_{z \in children(x)} f_{z \to x}(\pi_{FD(z)}(s', s)) \Bigg). \end{aligned}$$

and

$$\begin{aligned} s_t = \arg\max_{s' \in (\prod_{w \in BD(t)} S_w)} \Bigg(\sum_{v \in x \setminus y} u_v(\pi_{xnb_G(v)}(s', s)) + \\ \sum_{z \in children(x)} f_{z \to x}(\pi_{FD(z)}(s', s)) \Bigg). \end{aligned}$$

Moving to the second pass, in the first stage, elements of $BD(t)$ are assigned strategies according to s_t. Then, in the second stage, assuming that for $x \in X$ the whole path from $parent(x)$ to t is processed, elements from $s' \in BD(x) \setminus FD(x)$ are assigned according to $m_{x \to parent(x)}(\pi_{BD(x) \setminus FD(x)}(s))$, where s are states already assigned.

We provide the following theorem that shows the correctness of this procedure, i.e., that all functions are well defined and the procedure assigns strategies to all nodes in V. We show that the result of this assignment is indeed optimal in a separate theorem.

Theorem 1. *We have the following:*

1) *In the first stage of the first pass, if, for all $y \in children(x)$, functions $f_{y\to x}$ are known, then the computation of $f_{x\to parent(x)}$ is possible. For each $v \in x \setminus y$ we have $dom(u_v) \subset FD(x) \cup (BD(x) \setminus FD(x)) = BD(x) \cup FD(x)$ and for each $y \in children(x)$ we have $dom(f_{y\to x}) \subset BD(x) \cup FD(x)$.*
2) *Computation in the second stage of the second pass is possible, i.e., each $dom(u_v)$ is contained in nodes to which a strategy is already assigned or in nodes over which the max operator is calculated.*
3) *For each $y \in children(x)$, $dom(f_{y\to x})$ is contained in already set nodes and nodes over which the maximum is taken.*
4) *After the second pass all nodes have assigned strategies.*

Proof. First, we show that for $v \in x \setminus parent(x)$ we have $xnb_G(v) \subset BD(x) \cup FD(x)$. Consider $w \in xnb_G(v)$. If $w = v$ then obviously $w \in BD(x)$. If $w \in nb_G(v)$ then, by Lemma 1, $w \in x \cup \bigcup_{z\in offspring(x)} z$ and it obviously belongs to $xnb_G(x)$. Thus, by Remark 3, it also belongs to $BD(x)$. This gives us $w \in FD(x) \cup BD(x)$.

We also have, for all $z \in children(x)$, $FD(z) \subset BD(x)$. Thus, any element of the domain of $f_{z\to parent(z)} = f_{z\to x}$ is available when computing $f_{x\to y}$ according to the definition in the first pass of the algorithm. The same reasoning applies to $m_{x\to y}$ as this is just the selection of $\arg\max$ of $f_{x\to y}$.

Now, consider the second pass. The maxima can be calculated for the root, as according to the scheme functions and $f_{z\to t}$ for $z \in children(t)$ are known. Assume that assignments of the strategies are already done on the whole path from x to t. This means that it is a known argument for which the function $m_{z\to x}$ must be computed. The computation of this function extends the range over which the global solution is known. We must show that this extension is consistent among children of x. This is the case, however, where pieces of the solution computed for different children do not overlap. More formally, for $y_1, z_2 \in children(x)$ such that $z_1 \neq z_2$ we have $(BD(z_1) \setminus FD(z_1)) \cap (BD(z_2) \setminus FD(z_2)) = \emptyset$. Assume that this is not true, i.e., there exists $w \in (BD(z_1) \setminus FD(z_1)) \cap (BD(z_2) \setminus FD(z_2))$. We have an $w \in BD(z_1)$ that is either in z_1 or in one of its offspring. The same is true for z_2. From the definition of the tree, there is a path between any z_1 and any of its offspring to z_2 and any of its offspring leading through z_1, x, and z_2, as x is the common parent of z_1 and z_2. From the definition of tree decomposition we then must have $w \in z_1$, $w \in z_2$ and $w \in x$. This means that $w \in FD(z_1)$ and $w \in FD(z_2)$, which leads to a contradiction and concludes the proof. □

Now, we prove the correctness of the scheme in the case when domains S_v are finite.

Theorem 2. *For $(X, F) = T \in TD(G)$ and $t \in X$ being the root, and a set of functions $u_v : \prod_{xnb_G(v)} S_v \to \mathbb{R}$ where S_v is finite, the scheme finds a (global) maximum of the welfare function:*

$$wlf(x) = \sum_{v\in V} u_v(\pi_{xnb_G(v)}(x)).$$

Proof. Assume that $s \in \prod_{v\in V} S_v$ is the optimal assignment. In the second pass, the algorithm must consider it as part of the calculation of the operators max and $\arg\max$ in root assignment $\pi_{BD(t)}(s)$. □

IV. Towards a Distributed Implementation

We now outline an implementation of the algorithm presented in the previous section in a system where computations can be performed in a distributed manner. Assume that interacting nodes are organized in a way that reflects the structure of both graphs: the original interaction graph $G = (V, E)$ of the game and the chosen tree decomposition $(X, F) = T \in TD(G)$. Further, assume that each v is associated with a device. Devices can communicate with one another. Each device frequently decides about its action (strategy) from the set S_v and operates according to that choice. The result measured with numerical utility depends on the choice made by other devices associated with nodes in $nb_G(v)$. This utility function is denoted by u_v. Each device also knows the dependency of u_v from a combination of its own choice and the choices of other devices it depends on. This knowledge can be provided to the device or, in a more realistic scenario, it may come from empirical extrapolation of monitoring data. In the latter case, devices derive an empirical approximation of u_v from the observations of the actions of other devices, the knowledge of its own action, and the received utility[1]. Further, assume that for each $x \in X$ there is a distinguished device $v \in x$ which is further called the computation node and denoted by $cn(x)$. As the nodes in X (which are sets) are not disjoint, it is a valid situation where $cn(x) = cn(y)$ for $x \neq y$. The computation node is responsible for computing functions $f_{x\to parent(x)}$. Nodes for which u_v needs to be known directly by $cn(x)$, send to $cn(x)$ data which allows to compute u_v or its approximation, while the computation nodes $c(y)$ for $y \in children(x)$ send data that allows to compute the appropriate $f_{y\to x}$.

At some point (e.g., periodically) the following process occurs. Functions $f_{x\to parent(x)}$ and $m_{x\to parent(x)}$ are computed in the node $cn(x)$ for all x being leaves of T. Then, information sufficient to compute the function (or a sufficiently good approximation of the function) $f_{xparent(x)}$ and $m_{x\to parent(x)}$ is sent to $cn(parent(x))$. Each node not being the root, after receiving data from all its children, performs computation of $f_{x\to parent(x)}$ and $m_{x\to parent(x)}$ and sends information to computation node of the parent, etc. Finally, the computation node of the root t receives all information from computation nodes of its children and nodes it knows directly and itself performs computation corresponding to stage two of the first pass of the scheme. Next, the root sends information about m_t to all the nodes for which it determined an optimal strategy

[1]Such an empirical model easily captures the elements of randomness of the environment with statistical modeling.

and for all children sends $m_{x\to t}(m_t)$ to the computational node of child x. Then, for each x, the computational node that receives $m_{x\to parent(x)}(s)$ for an appropriate s sets the data in appropriate nodes and continues the process by sending appropriate data to its children. Upon receiving information about its assigned strategy, the node starts to apply it. After dissemination of the state across all nodes, the network should work in a new optimal regime.

V. INTERFERENCE GAME

We apply the proposed scheme to an interference game in a cellular network with downlink transmissions. To simplify the terminology, we identify BSs with a single cell. A set of these BSs is denoted by C. We assume that all stations use the same carrier and that downlink transmissions from BSs happen synchronously at discrete times. These assumptions are well justified for a large set of configurations of OFDM networks, e.g., for LTE TDD (Time Division Duplex) and NR networks.

For each discrete transmission opportunity, a BS decides if it transmits to at least one of the receivers assigned to it (UEs) or does not transmit at all. Again, to simplify we limit ourselves only to transmissions carrying user data, omitting in the model obligatory transmissions of reference and synchronisation signals, broadcast channels, etc. We also leave aside details of scheduling, i.e., to which of the devices the BS transmits.

In this setting, at each downlink transmission opportunity, the network is considered a game and each BS $c \in C$ is a player. With each player, we associate the strategy space $S_c = \{0, 1\}$, where 0 means that the BS is not transmitting and 1 means that it is transmitting. The payoff $u_c : \prod_{d\in C} S_d \to \mathbb{R}$ is defined as:

$$u_c(s) = \mathbb{E}[Tput_c(s)],$$

where $\mathbb{E}$ is the expected value and $Tput_c$ represents the downlink throughput and can be treated as a random variable depending on the selection of the receivers by the scheduling algorithm (so also based on the arriving data, what can be modeled as a stochastic process) and on activities of other BSs which are treated as interference. In general, the function of throughput may be complicated so, to pose the problem more concretely, we assume that if there is a transmission at all only one receiver is chosen for transmission at a given opportunity t and the function is of the form

$$Tput_c(s) = log_2\left(1 + \frac{Ps_c d(c, ue)^{-\theta}}{\sum\limits_{w\in C\setminus\{c\}} Ps_w d(w, ue)^{-\theta} + \nu}\right),$$

where the ue is a random position representing the receiver (UE) selected by c, $d(x, y)$ is the Euclidean distance between BS x and ue y , $\theta > 2$ is a propagation coefficient describing the attenuation of the signal with distance, ν is random noise and the expected value in u_c is taken over the distribution over the choice of ue.

We neglect the interference from far away stations, stations placed with an attenuating factor in between (e.g., walls), and stations that are close, but there are no receivers in the area where their interference is strong. The activity of these stations contributes to the interference term but this contribution is limited. This corresponds exactly to the elimination of the subset of players for which the influence function described in Section II-C is small. In this approximation, the sum in the denominator in the formula for u_c may be taken over the proper subset of $C \setminus \{c\}$. In other words, this leads to the assumption that the game has a graphical representation that is sparse.

We also assume that nodes exchange information about transmission allocation with their neighbours. That is, if a node strongly interferes with a given one, so that both are linked in the graph representing the game, it frequently sends information about its activity. This allows each node to estimate their u_c. Note that this information does not need to be exchanged in each cycle, which would be technically infeasible, but rather communicated in larger batches. To realize the scheme in a single cycle (i.e., when batches are sent and received, the amount node x must send and receive is $2 \times |nb_G(x)|$ messages. The length of the messages is proportional to the interval between subsequent exchanges counted in cycles as the information contains sequences of strategies used and payoffs received during the cycles.

Next, we move to the game in mixed strategies. In this case, the strategy space (we abuse notation and use the same letters for the new game) $S_c = [0, 1]$, and the choice of the strategy is a choice of the probability for transmission. Writing $u_{\Delta,c}$ explicitly leads us to the formula:

$$u_{\Delta,c}(x) = \sum_{s\in \prod\limits_{\{c\}\cup nb(c)} S_p} \left(\prod_{j\in\{c\}\cup nb(c)} x_j(s_j)\right) u_i(s).$$

Here we already incorporated into the formula the dependency of $u_{\Delta,c}$ only on its neighbours in the graphical representation. To restrict the domain of the utility functions to a finite set instead of the interval $[0, 1]$, for station c we restrict it to the subset $S_c = \{p_0, p_1\} \subset [0, 1]$. Finally, the game over which we optimize welfare is $(C, \{S_c\}_{c\in C}, \{u_c\}_{\Delta,c})$.

Assuming that the treewidth of the graphical representation is small and we can choose a proper tree decomposition T rooted in t, we apply the arrangement we prepared in the previous sections to calculate the maximum effectiveness of the game in terms of welfare. What needs to be calculated for node x is

$$f_{x\to parent(x)}(s) = \max_{s'\in BD(x)\setminus FD(x)} \left(\sum_{v\in x\setminus parent(x)} u_{\Delta,v}(\pi_v(s', s)) + \sum_{z\in children(x)} f_{z\to x}(\pi_{FD(z)}(s', s))\right).$$

As functions $f_{z\to x}$ are supposed to be known (and communicated to $cn(x)$, cf. Section IV), what remains to be computed is the maximum. The number of messages exchanged between $cn(x)$ in each step of the algorithm is proportional to $|nb_T(x)|$. The length of the message from $cn(x)$ to $cn(parent(x)$ in the

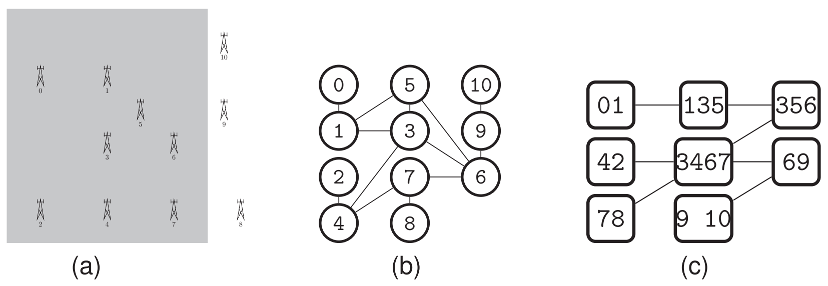

Fig. 2. Simulation topology: (a) BS arrangement, (b) corresponding graph, (c) corresponding tree. UEs are placed in the shaded area.

first stage of the algorithm is bounded by $|S|^{(|BD(x)\setminus FD(x)|}$, where $S = \max \_c \in C|S_c|$. This information, however, is passed between nodes once per single optimization run.

In this particular case, any computations of the maximum may be done by brute force. As a result, the network finds the best, from the global network's operation perspective, assignment of $\{p_c\}_{c\in C}$. Now, each node c in each moment independently performs a random decision if it should transmit (with probability p_c) or not (with probability $1 - p_c$). The whole process is repeated after a time reflecting the dynamic situation in the network.

The amount of brute force computation required depends on the treewidth, the structure of T . For networks with good structural properties , the computation may be feasible for even small computing units.

In the particular case discussed in this section, where we start from the interference game with two strategies for each player, then move to the mixed strategy game, and finally approximate the space of mixed strategies by discrete subsets, one can avoid brute force maximization over all possible combinations of strategies. The mixed strategy spaces, in this case, are one-dimensional and the payoff is monotonic with respect to each "mixed strategy variable" and allows maximization over each variable separately.

VI. Simulations

We evaluate the proposed algorithm by simulating the network topology of Fig. 2. We use a random and uniform distribution of 200 UEs within the grey-marked area. Each UE is assigned to the closest BS. BSs 7, 8 and 9, which are outside the grey rectangle, have all UEs assigned to them on the cell edge. For this arrangement, we measure the performance of the simulated network where BSs are busy 95% of the time. Then we run the same simulation, but with the optimisation algorithm enabled after 1000 steps. We restrict the choice of the BS time occupancy algorithm to two values: original 95% activity or decreased to 20%.

The results are presented in Fig. 3. The Y-axis is the average throughput over the whole history of the simulation, while the X-axis is the number of cycles since initializing the simulation. The average over history means that for cycle T we calculate $\frac{1}{T}\sum_{t=1}^{T} AvgTput(t)$, where $AvgTput(t)$ is the network throughput per UE in cycle t. The algorithm decides to decrease the activity of BSs 1, 3, 8, and 9 which results in an increase in overall network throughput compared to the situation when all BSs were busy. The increase is, however, in this setting relatively small (less than 5%). We conclude that the algorithm finds an optimal solution (subject to Theorem 2) under the proposed conditions.

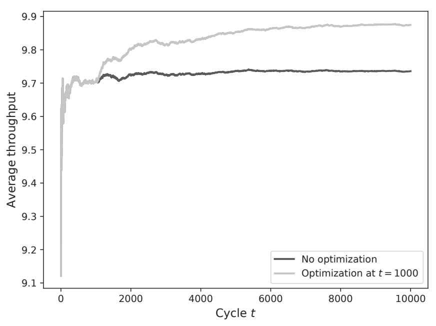

Fig. 3. Performance in the static scenario.

The previous example was static, i.e., the distribution of UE was constant and they were immobile. We obtain more interesting results for a changing environment. We add mobility to the initially randomly distributed UEs which are always assigned to the closest BS. Each UE moves on a piecewise linear trajectory with constant speed. We start from the random distribution in the same grey-marked area from Fig. 2a. Velocity changes only by changing direction which occurs when the device reaches a boundary of the rectangle, which is bigger than the initial area where we distributed UEs, and the change in direction follows a "billiard ball" rule.

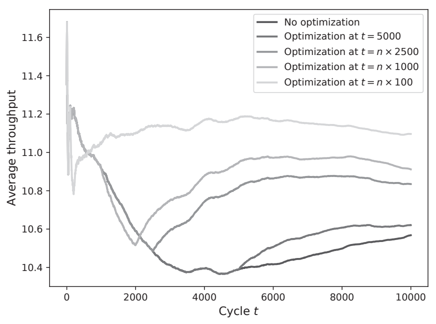

Fig. 4. Performance in the mobile scenario.

We check how the system performance changes when we apply the algorithm at various discrete moments: every 5000, 2500, 1000, and 100 cycles (Fig. 4). For “no optimization”, the only change in the environment is due to UE mobility and we observe throughput to initially decrease and later increase. Meanwhile, for the optimisation cases, throughput immediately improves and, as expected, more frequent optimisation leads to better performance. An interesting phenomenon is, however, observed for the initial parts of the curves corresponding to 100 and 1000 where we see significant initial degradation. This effect seems to be contrary to the proven fact that the algorithm optimizes throughput. Two factors contribute to this behaviour. First, when the environment changes sufficiently fast, the optimisation performed at a given moment does not necessarily optimize the system for future cycles. The mobility model is such that the initial concentration of UE in the early stage of the simulation is replaced, due to the ergodicity of the movement, by uniform random distribution in the late phase. So, we expect that the fastest changing phase of the evolution of the environment is the initial one. Second, the BSs in this example learn about the dependency of the interference between them based on current activity. Therefore, initially, when the number of samples is small, the estimation of the influence of interference may be incorrect. As soon as the number of samples gives statistically sound estimations, the system starts to optimize the empirically determined function which correctly captures the influence of interference.

VII. Conclusions and Future Work

We have presented a dynamic programming method for the computation of the sum of the welfare in a graphical game assuming knowledge of the tree decomposition of the underlying graph. We also have shown how the method can optimize cellular networks modeled as graphical games where the arising conflict between players (base stations) comes from the interference between them. In the proposed algorithm, we have omitted the analysis of the influence of important parameters and factors on the quality of the solutions found by the methods. These parameters include the cut-off threshold for influence between the nodes leading to sparse graphical game representation and the size and structure of the space S_c that approximates the full space of mixed strategies $[0, 1]$ in Section V. Other important factors also not discussed here are changing demand for traffic from individual UEs, the impact of uncertainty about the utility function, and more sophisticated strategies for limiting transmission. A detailed analysis of these aspects is the subject of ongoing investigations.

We have also focused on the optimisation of a simple network operation mode, i.e., we attempt to optimize welfare over a subset of mixed strategies. More sophisticated schemes, where the non-trivial correlation between nodes is involved, can be also solved by this type of algorithm.

References

[1] M. S. Ali, E. Hossain, and D. I. Kim, “Coordinated multipoint transmission in downlink multi-cell noma systems: Models and spectral efficiency performance,” *IEEE Wireless Communications*, vol. 25, no. 2, pp. 24–31, 2018. DOI: 10.1109/MWC.2018.1700094

[2] U. Madhow, D. R. Brown, S. Dasgupta, and R. Mudumbai, “Distributed massive mimo: Algorithms, architectures and concept systems,” in *2014 Information Theory and Applications Workshop (ITA)*, 2014. pp. 1–7. DOI: 10.1109/ITA.2014.6804225

[3] H. Garmani, D. Ait Omar, M. El Amrani, M. Baslam, and M. Jourhmane, “Joint beacon power and beacon rate control based on game theoretic approach in vehicular ad hoc networks,” *Infocommunications Journal*, vol. 13, no. 1, pp. 58–67, 2021. DOI: 10.36244/ICJ.2021.1.7

[4] G. Hollósi, C. Lukovszki, M. Bancsics, and G. Magyar, “Traffic swarm behaviour: Machine learning and game theory in behaviour analysis,” *Infocommunications Journal*, vol. 13, no. 4, pp. 19–27, 2021. DOI: 10.36244/ICJ.2021.4.3

[5] A. Lew and H. Mauch, Dynamic Programming: A Computational Tool (*Studies in Computational Intelligence*). Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 3540370137

[6] M. J. Kearns, M. L. Littman, and S. P. Singh, “Graphical models for game theory,” in *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, ser. UAI ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. ISBN 1558608001 pp. 253–260.

[7] H. L. Bodlaender, “Dynamic programming on graphs with bounded treewidth,” in *Automata, Languages and Programming*, T. Lepistö and A. Salomaa, Eds. Springer, 1988. ISBN 978-3-540-39291-0 pp. 105–118.

[8] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, 1st ed. Springer Publishing Company, Incorporated, 2016. ISBN 1447171640

[9] N. Robertson and P. Seymour, “Graph minors. ii. algorithmic aspects of tree-width,” *Journal of Algorithms*, vol. 7, no. 3, pp. 309–322, 1986. DOI: 10.1016/0196-6774(86)90023-4

[10] R. Diestel, *Graph Theory*, ser. Electronic library of mathematics. Springer, 2006. ISBN 9783540261834

[11] H. L. Bodlaender, “A linear time algorithm for finding tree-decompositions of small treewidth,” in *ACM Symposium on Theory of Computing*, ser. STOC ’93. New York, NY, USA: ACM, 1993. ISBN 0897915917 pp. 226–234. DOI: 10.1145/167088.167161.

[12] J. Matousek and R. Thomas, “Algorithms finding tree-decompositions of graphs,” *J. Algorithms*, vol. 12, no. 1, pp. 1–22, 1991. DOI: 10.1016/0196-6774(91)90020-Y

[13] E. Amir, “Approximation algorithms for treewidth,” *Algorithmica*, vol. 56, no. 4, pp. 448–479, 2010. DOI: 10.1007/s00453-008-9180-4

[14] H. L. Bodlaender, “Atourist guide through treewidth,” *Acta Cybernetica*, vol. 11, pp. 1–23, 1993.

[15] B. Li, “Tree decompositions and routing problems. (décompositions arborescentes et problèmes de routage),” Ph.D. dissertation, University of Nice Sophia Antipolis, France, 2014.

Artur Popławski received his M.Sc. degree in mathematics from Jagiellonian University in 2000. He is currently working at Nokia Solutions and Networks as a specification engineer on 4G/5G radio resource management.

Szymon Szott received his Ph.D. degree in telecommunications from AGH University in 2011, where he works as an associate professor. His professional interests are related to wireless networks (channel access, QoS provisioning, security). He is the author of over 70 research papers.