Joint Base Station Scheduling[⋆]

Thomas Erlebach[1], Riko Jacob[2], Matúš Mihaľák[3], Marc Nunkesser[2], Gábor Szabó[2], and Peter Widmayer[2]

[1] Department of Computer Science, University of Leicester
t.erlebach@mcs.le.ac.uk
[2] Department of Computer Science, ETH Zürich
{jacob, nunkesser, szabog, widmayer}@inf.ethz.ch
[3] Department of Information Technology and Electrical Engineering, ETH Zürich
mihalak@tik.ee.ethz.ch

Abstract. Consider a scenario where base stations need to send data to users with wireless devices. Time is discrete and slotted into synchronous rounds. Transmitting a data item from a base station to a user takes one round. A user can receive the data item from any of the base stations. The positions of the base stations and users are modeled as points in Euclidean space. If base station b transmits to user u in a certain round, no other user within distance at most $\|b-u\|_2$ from b can receive data in the same round due to interference phenomena. The goal is to minimize, given the positions of the base stations and users, the number of rounds until all users have their data.

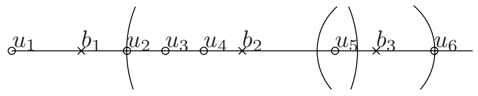
We call this problem the Joint Base Station Scheduling Problem (JBS) and consider it on the line (1D-JBS) and in the plane (2D-JBS). For 1D-JBS, we give a 2-approximation algorithm and polynomial optimal algorithms for special cases. We model transmissions from base stations to users as arrows (intervals with a distinguished endpoint) and show that their conflict graphs, which we call arrow graphs, are a subclass of the class of perfect graphs. For 2D-JBS, we prove *NP*-hardness and discuss an approximation algorithm.

1 Introduction

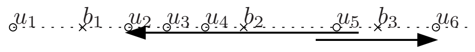
We consider different combinatorial aspects of problems that arise in the context of load balancing in time division networks. These problems turn out to be related to interval scheduling problems and interval graphs.

The general setting is that users with mobile devices are served by a set of base stations. In each time slot (round) of the time division multiplexing each base station serves at most one user. Traditionally, each user is assigned to a single base station that serves him until he leaves its cell or his demand is satisfied. The amount of data that a user receives depends on the strength of the signal

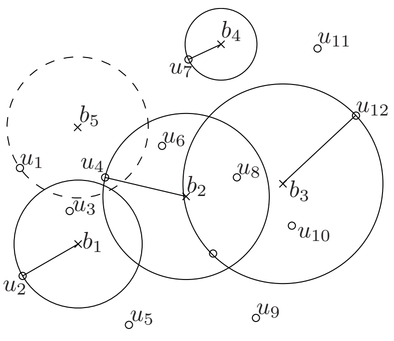
[⋆] Research partially supported by TH-Project TH-46/02-1 (Mobile phone antenna optimization: theory, algorithms, engineering, and experiments).



(a) This figure describes a possible situation in some time slot (round). Base station b_2 serves user u_2, b_3 serves user u_6. Users u_3, u_4 and u_5 are blocked and cannot be served. Base station b_1 cannot serve u_1 because this would create interference at u_2



(b) Arrow representation of 1(a)



(c) A possible situation in some time slot in the 2D case. Users u_2, u_4, u_7 and u_{12} are served. Base station b_5 cannot serve user u_1 here, because this would create interference at u_4 as indicated by the dashed circle

Fig. 1.1. The JBS-problem in one and two dimensions

that he receives from his assigned base station and on the interference, i.e. all signal power that he receives from other base stations. In [4], Das et al. propose a novel approach: Clusters of base stations jointly decide which users they serve in which round in order to increase network performance. Intuitively, this approach increases throughput, when in each round neighboring base stations try to serve pairs of users such that the mutual interference is low. We turn this approach into a discrete scheduling problem in one and two dimensions (see Figure 1.1), the Joint Base Station Scheduling problem (JBS).

In one dimension (see Figure 1.11(a)) we are given a set of n users as points $\{u_1, \ldots, u_n\}$ on a line and we are given positions $\{b_1, \ldots, b_m\}$ of m base stations. Note that such a setting could correspond to a realistic scenario where the base stations and users are located along a straight road. In our model, when a base station b_j serves a user u_i this creates interference in an interval of length $2|b_j - u_i|$ around the midpoint b_j. In each round each base station can serve at most one user such that at the position of this user there is no interference from any other base station. The goal is to serve all users in as few rounds as possible. In two dimensions (see Figure 1.11(c)), when base station b_j serves user u_i this creates interference in a disk with radius $\|b_j - u_i\|_2$ and center b_j.

The one-dimensional problem is closely related to interval scheduling problems, except that the particular way how interference operates leads to directed intervals (arrows). For these we allow that their tails can intersect (intersecting tails correspond to interference that does not affect the users at the heads of the arrows). We present results on this special interval scheduling problem. Similarly, the problem is related to interval graphs, except that we have conflict graphs of arrows together with the conflict rules defined by the interference (arrow graphs).

The problem of scheduling data transmissions in the smallest number of discrete rounds can be expressed as the problem of coloring the corresponding arrow graph with the smallest number of colors, where the colors represent rounds. In this paper, we prove that arrow graphs are perfect and can be colored optimally in $\mathcal{O}(n \log n)$ time. For the one-dimensional JBS problem with evenly spaced base stations we give a polynomial-time dynamic programming algorithm. For the general one-dimensional JBS problem, we show that for any fixed k the question whether all users can be served in k rounds can be solved in polynomial time. From the perfectness of arrow graphs and the existence of a polynomial-time algorithm for computing maximum weighted cliques in these graphs we derive a 2-approximation algorithm for JBS based on an LP relaxation and rounding. For the two-dimensional JBS problem, we show that it is *NP*-complete, and we discuss an approximation algorithm for a constrained version of the problem.

1.1 Related Work

Das et al. [4] propose an involved model for load balancing that takes into account different fading effects and calculates the resulting signal to noise ratios at the users for different schedules. In each round only a subset of all base stations is used in order to keep the interference low. The decision which base stations to use is taken by a central authority. The search for this subset is formulated as a (nontrivial) optimization problem that is solved by complete enumeration and that assumes complete knowledge of the channel conditions. The authors perform simulations on a hexagonal grid, propose other algorithms, and reach the conclusion that the approach has the potential to increase throughput.

There is a rich literature on interval scheduling and selection problems (see [6, 12] and the references given there for an overview). Our problem is more similar to a setting with several machines where one wants to minimize the number of machines required to schedule all intervals. A version of this problem where intervals have to be scheduled within given time windows is studied in [3]. Inapproximability results for the variant with a discrete set of starting times for each interval are presented in [2].

1.2 Problem Definitions and Model

We fully define the problems of interest in this section. Throughout the paper we use standard graph-theoretic terminology, see e.g. [14]. In the one-dimensional case we are given positions of base stations $B = \{b_1, \dots, b_m\}$ and users $U = \{u_1, \dots, u_n\}$ on the line in left-to-right order. Conceptually, it is more convenient to think of the interference region that is caused by some base station b_j serving a user u_i as an *interference arrow* of length $2|b_j - u_i|$ with midpoint b_j pointing to the user as shown in Figure 1.11(b). The interference arrow for the pair (u_i, b_j) has its head at u_i and its midpoint at b_j. We denote the set of all arrows resulting from pairs $P \subseteq U \times B$ by $\mathcal{A}(P)$. If it is clear from the context, we call the interference arrows just *arrows*. If two users are to be scheduled in the same round, then each of them must not get any interference from any other base station. Thus, two arrows are *compatible* if no head is contained in the other arrow; otherwise, we say that they are in *conflict*. (Formally, the head u_i of the arrow for (u_i, b_j) is contained in the arrow for (u_j, b_k) if u_i is contained in the closed interval $[b_k - |u_j - b_k|, b_k + |u_j - b_k|]$.) If we want to emphasize which user is affected by the interference from another transmission, we use the term *blocking*, i.e. arrow a_i blocks arrow a_j if a_j's head is contained in a_i. For each user we have to decide from which base station she is served. This corresponds to a selection of an arrow for her. Furthermore, we have to decide in which round each selected arrow is scheduled under the side constraint that all arrows in one round must be compatible. For this purpose it is enough to label the arrows with colors that represent the rounds.

For the two-dimensional JBS problem we have positions in $\mathbb{R}^2$ and *interference disks* $d(b_i, u_j)$ with center b_i and radius $\|b_i - u_j\|_2$ instead of arrows. We denote the set of interference disks for the user base-station pairs from a set P by $\mathcal{D}(P)$. Two interference disks are in conflict if the user that is served by one of the disks is contained in the other disk; otherwise, they are compatible. The problems can now be stated as follows:

1D-JBS

Input: A set of user positions $U = \{u_1, \ldots, u_n\} \subset \mathbb{R}$ and base station positions $B = \{b_1, \ldots, b_m\} \subset \mathbb{R}$.

Output: A set P of n user base-station pairs such that each user is in exactly one pair, and a coloring $C : \mathcal{A}(P) \to \mathbb{N}$ of the set $\mathcal{A}(P)$ of corresponding arrows such that any two arrows $a_i, a_j \in \mathcal{A}(P),\ a_i \neq a_j$, with $C(a_i) = C(a_j)$ are compatible.

Objective: Minimize the number of colors used.

2D-JBS

Input: A set of user positions $U = \{u_1, \ldots, u_n\} \subset \mathbb{R}^2$ and base station positions $B = \{b_1, \ldots, b_m\} \subset \mathbb{R}^2$.

Output: A set P of n user base-station pairs such that each user is in exactly one pair, and a coloring $C : \mathcal{D}(\mathcal{P}) \to \mathbb{N}$ of the set $\mathcal{D}(\mathcal{P})$ of corresponding disks such that any two disks $d_i, d_j \in \mathcal{D}(\mathcal{P}),\ d_i \neq d_j$, with $C(d_i) = C(d_j)$ are compatible.

Objective: Minimize the number of colors used.

From the problem definitions above it is clear that both the 1D- and the 2D-JBS problems consist of a *selection problem* and a *coloring problem*. In the selection problem we want to select one base station for each user in such a way that the arrows (disks) corresponding to the resulting set P of user base-station pairs can be colored with as few colors as possible. We call a selection P *feasible* if it contains exactly one user base-station pair for each user. Determining the cost of a selection is then the coloring problem. This can also be viewed as a problem in its own right, where we no longer make any assumption on how the set of arrows (for the 1D problem) is produced. The conflict graph $G(A)$ of a set A of arrows is the graph in which every vertex corresponds to an arrow and there is an edge between two vertices if the corresponding arrows are in conflict. We call such conflict graphs of arrows *arrow graphs*. The *arrow graph coloring problem* asks for a proper coloring of such a graph. It is similar in spirit to the coloring of interval graphs. As we will see in Section 2.1, the arrow graph coloring problem can be solved in time $\mathcal{O}(n \log n)$. We finish this section with a simple lemma that leads to a definition:

Lemma 1. *For each 1D-JBS instance there is an optimal solution in which each user is served either by the closest base station to his left or by the closest base station to his right.*

Proof. This follows by a simple exchange argument: Take any optimal solution that does not have this form. Then exchange the arrow where a user is not served by the closest base station in some round against the arrow from the closest base station on the same side (which must be idle in that round). Shortening an arrow without moving its head can only resolve conflicts. Thus, there is also an optimal solution with the claimed property. □

The two possible arrows by which a user can be served according to this lemma are called *user arrows*. It follows that for a feasible selection one has to choose one user arrow from each pair of user arrows.

2 Case on the Line—1D-JBS

As mentioned above, solving the 1D-JBS problem requires selecting an arrow for each user and coloring the resulting arrow graph with as few colors as possible.

2.1 Relation of Arrow Graphs to Other Graph Classes

In order to gain a better understanding of arrow graphs, we first discuss their relationship to other known graph classes.[1] We refer to [1, 13] for definitions and further information about the graph classes mentioned in the following.

First, it is easy to see that arrow graphs are a superclass of interval graphs: Any interval graph can be represented as an arrow graph with all arrows pointing in the same direction.

An arrow graph can be represented as the intersection graph of triangles on two horizontal lines $y = 0$ and $y = 1$: Simply represent an arrow with left endpoint ℓ and right endpoint r that points to the right (left) as a triangle with

[1] The connections between arrow graphs and known graph classes such as PI* graphs, trapezoid graphs, co-comparability graphs, AT-free graphs, and weakly chordal graphs were observed by Ekki Köhler, Jeremy Spinrad, Ross McConnell, and R. Sritharan at the seminar "Robust and Approximative Algorithms on Particular Graph Classes", held in Dagstuhl Castle during May 24–28, 2004.

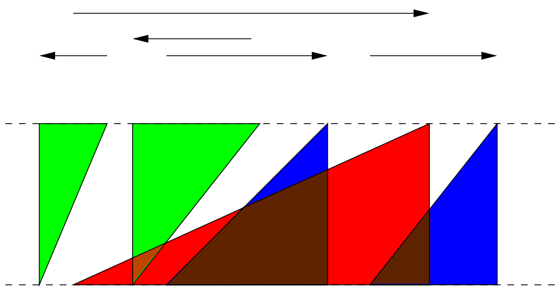


Fig. 2.1. An arrow graph (top) and its representation as a PI* graph (bottom)

corners $(\ell, 0)$, $(r, 0)$, and $(r, 1)$ (with corners $(r, 1)$, $(\ell, 1)$, and $(\ell, 0)$). It is easy to see that two triangles intersect if and only if the corresponding arrows are in conflict. See Figure 2.1 for an example.

Intersection graphs of triangles with endpoints on two parallel lines are called PI* graphs. They are a subclass of trapezoid graphs, which are the intersection graphs of trapezoids that have two sides on two fixed parallel lines. Trapezoid graphs are in turn a subclass of co-comparability graphs, a well-known class of perfect graphs. Therefore, the containment in these known classes of perfect graphs implies the perfectness of arrow graphs. Consequently, the size of a maximum clique in an arrow graph always equals its chromatic number.

As arrow graphs are a subclass of trapezoid graphs, we can apply known efficient algorithms for trapezoid graphs to arrow graphs. Felsner et al. [7] give algorithms with running-time $\mathcal{O}(n \log n)$ for chromatic number, weighted independent set, clique cover, and weighted clique in trapezoid graphs with n vertices, provided that the trapezoid representation is given. Their algorithm for chromatic number leads to a simple greedy coloring algorithm for arrow graphs (see [5]).

We sum up the discussed properties of arrow graphs in the following theorem.

Theorem 1. *Arrow graphs are perfect. In arrow graphs chromatic number, weighted independent set, clique cover, and weighted clique can be solved in time* $\mathcal{O}(n \log n)$.

One can also show that arrow graphs are AT-free (i.e., do not contain an asteroidal triple) and weakly chordal.

2.2 1D-JBS with Evenly Spaced Base Stations

Now we consider the 1D-JBS problem under the assumption that the base stations are evenly spaced. We are given m base stations $\{b_1, \ldots, b_m\}$ and n users $\{u_1, \ldots, u_n\}$ on a line, where the distance between any two neighboring base stations is the same. This assumption can be viewed as an abstraction of the fact that in practice, base stations are often placed in regular patterns and not in a completely arbitrary fashion.

Let d denote the distance between two neighboring base stations. The base stations partition the line into two *rays* and a set of *intervals* $\{v_1, \ldots, v_{m-1}\}$. In this section we additionally require that no user to the left of the leftmost base station be further away from it than distance d, and that the same holds for the right end. We define a solution to be *non-crossing* if there are no two users u and w in the same interval such that u is to the left of w, u is served from the right, and w from the left.

Lemma 2. *There is an optimal solution that is non-crossing.*

Proof. Take any optimal solution s that is not non-crossing. We show that such a solution can be transformed into another optimal solution s' that is non-crossing. Let u and w be two users such that u and w are in the same interval, u is to the left of w, and u is served by the right base station b_r in round t_1 by arrow a_r and w is served by the left base station b_l in round t_2 by arrow a_l; trivially, $t_1 \neq t_2$. Modify s in such a way that at t_1 base station b_r serves w and at t_2 base station b_l serves u. This new solution is still feasible because first of all both the left and the right involved arrows a_l and a_r have become shorter. This implies that both a_l and a_r can only block fewer users. On the other hand, the head of a_l has moved left and the head of a_r has moved right. It is impossible that they are blocked now because of this movement: In t_1 this could only happen if there were some other arrows containing w, the new head of a_r. This arrow cannot come from the left, because then it would have blocked also the old arrow. It cannot come from b_r because b_r is busy. It cannot come from a base station to the right of b_r, because such arrows do not reach any point to the left of b_r (here we use the assumption that the rightmost user is no farther to the right of the rightmost base station than d, and that the base stations are evenly spaced). For t_2 the reasoning is symmetric. $\square$

The selection of arrows in any non-crossing solution can be completely characterized by a sequence of $m-1$ *division points*, such that the i^{th} division point is the index of the last user that is served from the left in the i^{th} interval. (The case where all users in the i^{th} interval are served from the right is handled by choosing the i^{th} division point as the index of the rightmost user to the left of the interval, or as 0 if no such user exists.) A brute-force approach could now enumerate over all possible $\mathcal{O}(n^{m-1})$ division point sequences (*dps*) and color the selection of arrows corresponding to each dps with the greedy algorithm.

Dynamic Programming

We can solve the 1D-JBS problem with evenly spaced base stations more efficiently by a dynamic programming algorithm that runs in polynomial time. The idea of the algorithm is to consider the base stations and thus the intervals in left-to-right order. We consider the cost $\chi_i(d_{i-1}, d_i)$ of an optimal solution up to the ith base station conditioned on the position of the division points d_{i-1} and d_i in the intervals v_{i-1} and v_i, respectively, see Figure 2.2.

Definition 1. *We denote by $\chi_i(\alpha, \beta)$ the minimum number of colors needed to serve users u_1 to u_β using the base stations b_1 to b_i under the condition that base station b_i serves exactly users $u_{\alpha+1}$ to u_β and ignoring the users $u_{\beta+1}, \ldots, u_n$.*

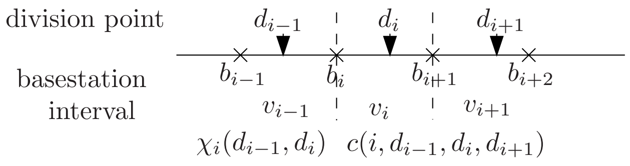


Fig. 2.2. Dynamic programming approach

Let $\Lambda(v_i)$ denote the set of potential division points for interval v_i, i.e., the set of the indices of users in v_i and of the rightmost user to the left of v_i (or 0 if no such user exists). The values $\chi_1(d_0, d_1)$ for $d_0 = 0$ (all users to the left of b_1 must be served by b_1 in any solution) and $d_1 \in \Lambda(v_1)$ can be computed directly by using the coloring algorithm from [7]. For $i \geq 1$, we compute the values $\chi_{i+1}(d_i, d_{i+1})$ for $d_i \in \Lambda(v_i)$, $d_{i+1} \in \Lambda(v_{i+1})$ from the table for $\chi_i(\cdot, \cdot)$. If we additionally fix a division point d_{i-1} for interval v_{i-1}, we know exactly which selected arrows intersect interval v_i regardless of the choice of other division points. Observe that this only holds for evenly spaced base stations and no "far out" users. For this selection, we can determine with the coloring algorithm from [7] how many colors are needed to color the arrows intersecting v_i. Let us call this number $c(i, d_{i-1}, d_i, d_{i+1})$ for interval v_i and division points d_{i-1}, d_i and d_{i+1}. We also know how many colors we need to color the arrows intersecting intervals v_0 to v_{i-1}. For a fixed choice of division points d_{i-1}, d_i and d_{i+1} we can combine the two colorings corresponding to $\chi_i(d_{i-1}, d_i)$ and $c(i, d_{i-1}, d_i, d_{i+1})$: Both of these colorings color all arrows of base station b_i, and these arrows must all have different colors in both colorings. No other arrows are colored by both colorings, so $\chi_i(d_{i-1}, d_i)$ and $c(i, d_{i-1}, d_i, d_{i+1})$ agree up to redefinition of colors. We can choose the best division point d_{i-1} and get

$$\chi_{i+1}(d_i, d_{i+1}) = \min_{d_{i-1} \in \Lambda(v_{i-1})} \max \{\chi_i(d_{i-1}, d_i), c(i, d_{i-1}, d_i, d_{i+1})\}$$

The running time is dominated by the calculation of the $c(\cdot)$ values. There are $\mathcal{O}(m \cdot n^3)$ such values, and each of them can be computed in time $\mathcal{O}(n \log n)$ using the coloring algorithm from [7]. The optimal solution can be found in the usual way by tracing back where the minimum was achieved from $\chi_m(x, n)$. Here the x is chosen among the users of the interval before the last base station such that $\chi_m(x, n)$ is minimum. For the traceback it is necessary to store in the computation of the χ values where the minimum was achieved. The traceback yields a sequence of division points that defines the selection of arrows that gives the optimal schedule. Altogether, we have shown the following theorem:

Theorem 2. *The base station scheduling problem for evenly spaced base stations can be solved in time $\mathcal{O}(m \cdot n^4 \log n)$ by dynamic programming.*

Note that the running time can also be bounded by $\mathcal{O}(m \cdot u_{\max}^4 \log u_{\max})$, where $u_{\max}$ is the maximum number of users in one interval.

2.3 Exact Algorithm for the k-Decision Problem

In this section we present an exact algorithm for the decision variant k-1D-JBS of the 1D-JBS problem: For given k and an instance of 1D-JBS, decide whether all users can be served in at most k rounds. We present an algorithm for this problem that runs in $\mathcal{O}(m \cdot n^{2k+1} \log n)$ time.

We use the result from Section 2.1 that arrow graphs are perfect. Thus the size of the maximum clique of an arrow graph equals its chromatic number.

The idea of the algorithm, which we call $A_{k-\mathrm{JBS}}$, is to divide the problem into subproblems, one for each base station, and then combine the partial solutions to a global one.

For base station b_i, the corresponding subproblem S_i considers only arrows that intersect b_i and arrows for which the alternative user arrow[2] intersects b_i. Call this set of arrows A_i. We call S_{i-1} and S_{i+1} *neighbors* of S_i. A solution to S_i consists of a feasible selection of arrows from A_i of cost no more than k, i.e. the selection can be colored with at most k colors. To find all such solutions we enumerate all possible selections that can lead to a solution in k rounds. For S_i we store all such solutions $\{s_i^1, \ldots, s_i^I\}$ in a table T_i. We only need to consider selections in which at most $2k$ arrows intersect the base station b_i. All other selections need more than k rounds, because they must contain more than k arrows pointing in the same direction at b_i. Therefore, the number of entries of T_i is bounded by $\sum_{j=0}^{2k} \binom{n}{j} = \mathcal{O}(n^{2k})$. We need $\mathcal{O}(n \log n)$ time to evaluate a single selection with the coloring algorithm from [7]. Selections that cannot be colored with at most k colors are marked as irrelevant and ignored in the rest of the algorithm. We build up the global solution by choosing a set of feasible selections $s_1, \ldots, s_m$ in which all neighbors are compatible, i.e. they agree on the selection of common arrows. It is easy to see that in such a global solution all subsolutions are pairwise compatible.

We can find such a set of compatible neighbors by going through the tables in left-to-right order and marking every solution in each table as *valid* if there is a compatible, valid solution in the table of its left neighbor, or as *invalid* otherwise. A solution s_i marked as valid in table T_i thus indicates that there are solutions $s_1, \ldots, s_{i-1}$ in $T_1, \ldots, T_{i-1}$ that are compatible with it and pairwise compatible. In the leftmost table T_1, every feasible solution is marked as valid. When the marking has been done for the tables of base stations $b_1, \ldots, b_{i-1}$, we can perform the marking in the table T_i for b_i in time $\mathcal{O}(n^{2k+1})$ as follows. First, we go through all entries of the table T_{i-1} and, for each such entry, in time $\mathcal{O}(n)$ discard the part of the selection affecting pairs of user arrows that intersect only b_{i-1} but not b_i, and enter the remaining selection into an intermediate table $T_{i-1,i}$. The table $T_{i-1,i}$ stores entries for all selections of arrows from pairs of user arrows intersecting both b_{i-1} and b_i. An entry in $T_{i-1,i}$ is marked as valid if at least one valid entry from T_{i-1} has given rise to the entry. Then, the entries of T_i are considered one by one, and for each such entry s_i the algorithm looks

[2] For every user there are only two user arrows that we need to consider (Lemma 1). If we consider one of them, the other one is the *alternative user arrow*.

up in time $\mathcal{O}(n)$ the unique entry in $T_{i-1,i}$ that is compatible with s_i to see whether it is marked as valid or not, and marks the entry in T_i accordingly. If in the end the table T_m contains a solution marked as valid, a set of pairwise compatible solutions from all tables exists and can be retraced easily.

The overall running time of the algorithm is $\mathcal{O}(m \cdot n^{2k+1} \cdot \log n)$. There is a solution to k-1D-JBS if and only if the algorithm finds such a set of compatible neighbors. In the technical report [5] we give a formal proof of this statement.

Theorem 3. *Problem k-1D-JBS can be solved in time* $\mathcal{O}(m \cdot n^{2k+1} \cdot \log n)$.

2.4 Approximation Algorithm

In this section we present an approximation algorithm for 1D-JBS that relies on the properties of arrow graphs from Theorem 1. Let A denote the set of all user arrows of the given instance of 1D-JBS. From the perfectness of arrow graphs it follows that it is equivalent to ask for a feasible selection $A_{\mathrm{sel}} \subseteq A$ minimizing the chromatic number of its arrow graph $G(A_{\mathrm{sel}})$ (among all feasible selections) and to ask for a feasible selection A_{sel} minimizing the maximum clique size of $G(A_{\mathrm{sel}})$ (among all feasible selections). Exploiting this equivalence, we can express the 1D-JBS problem as an integer linear program as follows. We introduce two indicator variables l_i and r_i for every user i that indicate whether she is served by the left or by the right base station, i.e. if the user's left or right user arrow is selected. Moreover, we ensure by the constraints that no cliques in $G(A_{\mathrm{sel}})$ are large and that each user is served. The ILP formulation is as follows:

$$\begin{aligned}
\min\ & k && (2.1)\\
\text{s.t. } & \sum_{l_i \in C} l_i + \sum_{r_i \in C} r_i \leq k \qquad \forall \text{ cliques } C \text{ in } G(A) && (2.2)\\
& l_i + r_i = 1 \qquad \forall i \in \{1, \ldots, |U|\} && (2.3)\\
& l_i, r_i \in \{0,1\} \qquad \forall i \in \{1, \ldots, |U|\} && (2.4)\\
& k \in \mathbb{N} && (2.5)
\end{aligned}$$

The natural LP relaxation is obtained by allowing $l_i, r_i \in [0,1]$ and $k \geq 0$. Given a solution to this relaxation, we can use a rounding technique to get an assignment of users to base stations that has cost at most twice the optimum, i.e., we obtain a 2-approximation algorithm. Let us denote by *opt* the optimum number of colors needed to serve all users. Then $opt \geq k$, because the optimum integer solution is a feasible fractional solution. Construct now a feasible solution from a solution to the relaxed problem by rounding $l_i := \lfloor l_i + 0.5 \rfloor$, $r_i := 1 - l_i$. Before the rounding the size of every (fractional) clique is at most k; afterwards the size can double in the worst case. Therefore, the cost of the rounded solution is at most $2k \leq 2opt$. We remark that there are examples where the cost of an optimal solution to the relaxed program is indeed smaller than the cost of an optimal integral solution by a factor of 2.

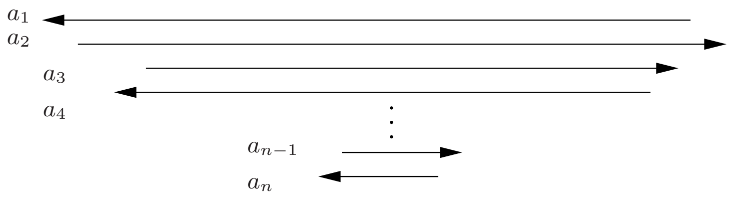


Fig. 2.3. Example of an arrow graph with an exponential number of maximum cliques. For every choice of arrows from a compatible pair (a_{2i-1}, a_{2i}) we get a clique of size $n/2$, which is maximum. The arrow graph can arise from a 1D-JBS instance with two base stations in the middle and $n/2$ users on either side

One issue that needs to be discussed is how the relaxation can be solved in time polynomial in n and m, as there can be an exponential number of constraints (2.2). (Figure 2.3 shows that this can really happen. The potentially exponential number of maximal cliques in arrow graphs distinguishes them from interval graphs, which have only a linear number of maximal cliques.) Fortunately, we can still solve such an LP in polynomial time with the ellipsoid method of Khachiyan [11] applied in a setting similar to [10]. This method only requires a separation oracle that provides us for any values of l_i, r_i with a violated constraint, if one exists. It is easy to check for a violation of constraints (2.3) and (2.4). For constraints (2.2), we need to check if for given values of l_i, r_i the maximum weighted clique in $G(A)$ is smaller than k. By Theorem 1 this can be done in time $\mathcal{O}(n \log n)$. Summarizing, we get the following theorem:

Theorem 4. *There is a polynomial-time* 2*-approximation algorithm for the 1D-JBS problem.*

3 General Case in the Plane—2D-JBS

We analyze the two-dimensional version (2D-JBS) of the base station scheduling problem. We show that the decision variant k-2D-JBS of the 2D-JBS problem is *NP*-complete and we present a constant factor approximation algorithm for a constrained version of it. The k-2D-JBS problem asks for a given k and an instance of 2D-JBS whether the users can be served in at most k rounds.

3.1 *NP*-Completeness of the 2D-JBS Problem

Here we briefly sketch our reduction from the general graph k-colorability problem [8] to 2D-JBS; the complete proof can be found in the technical report [5]. Our reduction follows the methodology presented in [9] for unit disk k-colorability.

Given any graph G, it is possible to construct in polynomial time a corresponding 2D-JBS instance that can be scheduled in k rounds if and only if G is k-colorable. We use an embedding of G into the plane which allows us to replace the edges of G with suitable base station chains with several users in a systematic way such that k-colorability is preserved. Our main result is the following:

Theorem 5. *The k-2D-JBS problem in the plane is NP-complete for any fixed $k \geq 3$.*

In the k-2D-JBS instances used in our reduction, the selection of the base station serving each user is uniquely defined by the construction. Hence, our reduction proves that already the coloring step of the 2D-JBS problem is *NP*-complete.

Corollary 1. *The coloring step of the k-2D-JBS problem is NP-complete for any fixed $k \geq 3$.*

3.2 Approximation Algorithms

Bounded Geometric Constraints. We consider instances where the base stations are at least a distance Δ from each other and have limited power to serve a user, i.e., every base station can serve only users that are at most $R_{\max}$ away from it. We also assume that for every user there is at least one base station that can reach the user. We present a simple algorithm achieving an approximation ratio depending only on the parameters Δ and $R_{\max}$.

Tiling the plane into a grid of squares of size $2R_{\max} \times 2R_{\max}$ and labelling the grid as in Figure 3.1 we get sets of squares S_a, S_b, S_c and S_d, where S_x is the set of squares with label x. We can place the grid in such a way that no base station lies on the boundary of a square. Note that if two base stations b_i and b_j are in different squares of the same label, their distance is greater than $2R_{\max}$ and, therefore, their transmissions cannot interfere. Now the algorithm proceeds as follows. While not all users are served, it goes in four steps through labels a, b, c and d. For each square of the current label, it repeatedly chooses an arbitrary base station from that square that can serve some user (i.e., the user

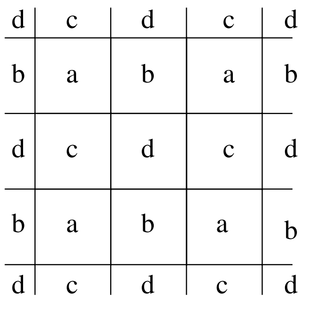


Fig. 3.1. Tiling of the plane into a grid of squares of size $2R_{\max} \times 2R_{\max}$ and labelling of the squares

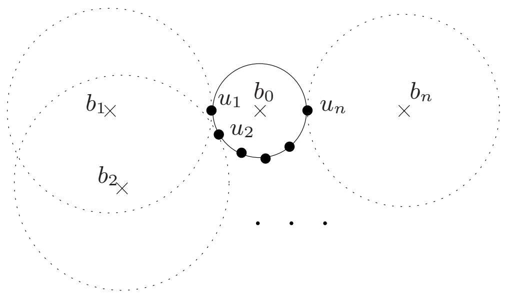


Fig. 3.2. A greedy approach serves n users placed on a common interference disk in n time steps. An optimum algorithm can serve the users in one time step by assigning u_i to base station b_i, which lies on a halfline determined by b_0 and u_i

is at distance at most $R_{\max}$ from the base station and there is no interference at the user in the current round) and schedules the transmission from the chosen base station to that user in the current round. It keeps choosing base stations in the current square in this way as long as possible. If, after executing this step for all squares of the current label, there are still some unserved users, the algorithm proceeds to the next label and starts a new round.

We analyze the algorithm as follows. For every grid square s, let k_s denote the number of rounds in which a base station in s serves a user (in the solution computed by the algorithm). Let u_s be the number of users served by base stations in s. Note that $u_s \geq k_s$. Choose a square s for which k_s is maximum. It is clear that the solution of the algorithm uses at most $4k_s$ rounds, since the squares with the label of s are considered at least once every four rounds. Now we derive a lower bound on the number of rounds in the optimum solution. The u_s users served by the algorithm from base stations in s are contained in a square with side length $4R_{\max}$, as the maximum transmission radius is $R_{\max}$. The base stations that the optimum solution uses to serve these users must then be contained in a square of side length $6R_{\max}$ for the same reason. As disks of radius $\Delta/2$ centered at different base stations are interior-disjoint by assumption, an easy area argument shows that there can be at most $\rho = (6R_{\max} + \Delta)^2/\pi(\Delta/2)^2$ base stations in such a square. Therefore, even the optimal algorithm cannot serve more than ρ of the u_s users in one round. Hence, the optimum solution needs at least k_s/ρ rounds. This establishes the following theorem.

Theorem 6. *There exists an approximation algorithm with approximation ratio* $\frac{16}{\pi}(\frac{6R_{\max}+\Delta}{\Delta})^2$ *for 2D-JBS in the setting where any two base stations are at least* Δ *away from each other and every base station can serve only users within distance at most* $R_{\max}$ *from it.*

General 2D-JBS. In the technical report [5] we also discuss lower bounds on three natural greedy approaches for the general 2D-JBS problem: serve a maximum number of users in each round (*max-independent-set*), or repeatedly choose an interference disk of an unserved user with minimum radius (*smallest-disk-first*), or repeatedly choose an interference disk containing the fewest other unserved users (*fewest-users-in-disk*). In [5] we prove the following theorem.

Theorem 7. *There are instances* (U, B) *of 2D-JBS in general position (i.e., with no two users located on the same circle centered at a base station) for which the maximum-independent-set greedy algorithm, the smallest-disk-first greedy algorithm, and the fewest-users-in-disk greedy algorithm have approximation ratio* $\Omega(\log n)$*, where* $n = |U|$*.*

For instances of 2D-JBS that are not in general position, the smallest-disk-first greedy algorithm can have approximation ratio n, as shown in Figure 3.2.

4 Conclusion and Open Problems

In this paper we study the 1D- and 2D-JBS problems that arise in the context of coordinated scheduling in packet data systems. These problems can be split into a selection and a coloring problem. In the one-dimensional case, we have shown that the coloring problem leads to the class of arrow graphs, for which we have discussed its relation to other graph classes and algorithms. For the selection problem we propose an approach based on LP relaxation and rounding. For the 2D-problem, we have shown its NP-completeness. Several problems remain unsolved. In particular, it is open whether the 1D-JBS problem is *NP*-complete. For 2D-JBS it would be interesting to design approximation algorithms whose approximation ratio does not depend on the ratio $\frac{R_{\max}}{\Delta}$. Moreover, algorithmic results for more refined models would be interesting.

References

1. A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: A survey.* SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
2. J. Chuzhoy and S. Naor. New hardness results for congestion minimization and machine scheduling. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing (STOC'04)*, pages 28–34, 2004.
3. M. Cielibak, T. Erlebach, F. Hennecke, B. Weber, and P. Widmayer. Scheduling jobs on a minimum number of machines. In *Proceedings of the 3rd IFIP International Conference on Theoretical Computer Science*, pages 217–230. Kluwer, 2004.
4. S. Das, H. Viswanathan, and G. Rittenhouse. Dynamic load balancing through coordinated scheduling in packet data systems. In *Proceedings of Infocom'03*, 2003.
5. T. Erlebach, R. Jacob, M. Mihaľák, M. Nunkesser, G. Szabó, and P. Widmayer. Joint base station scheduling. Technical Report 461, ETH Zürich, Institute of Theoretical Computer Science, 2004.
6. T. Erlebach and F. C. R. Spieksma. Interval selection: Applications, algorithms, and lower bounds. *Algorithmica*, 46:27–53, 2001.
7. S. Felsner, R. Müller, and L. Wernisch. Trapezoid graphs and generalizations, geometry and algorithms. *Discrete Applied Mathematics*, 74:13–32, 1997.
8. M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, 1979.
9. A. Gräf, M. Stumpf, and G. Weißenfels. On coloring unit disk graphs. *Algorithmica*, 20(3):277–293, March 1998.
10. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.
11. L. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.
12. F. C. R. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling*, 2:215–227, 1999.
13. J. P. Spinrad. *Efficient Graph Representations*, volume 19 of *Field Institute Monographs.* AMS, 2003.
14. D. West. *Introduction to Graph Theory.* Prentice Hall, 2nd edition, 2001.