Minimizing Flow Time in the Wireless Gathering Problem

VINCENZO BONIFACI
Max Planck Institute for Informatics
PETER KORTEWEG
TU Eindhoven
ALBERTO MARCHETTI-SPACCAMELA
Sapienza University of Rome
and
LEEN STOUGIE
VU University and CWI Amsterdam

We address the problem of efficient data gathering in a wireless network through multi-hop communication. We focus on two objectives related to flow times, that is, the times spent by data packets in the system: minimization of the maximum flow time and minimization of the average flow time of the packets. For both problems we prove that, unless $\mathbf{P} = \mathbf{NP}$, no polynomial time algorithm can approximate the optimal solution within a factor less than $\Omega(m^{1-\epsilon})$ for any $0 < \epsilon < 1$, where m is the number of packets. We then assess the performance of two natural algorithms by proving that their cost remains within the optimal cost of the respective problem if we allow the algorithms to transmit data at a speed 5 times higher than that of the optimal solutions to which we compare them.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*routing and layout; sequencing and scheduling*; C.2.1 [**Computer-Communication Networks**]: Network Protocols—*routing protocols*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms; network problems*

General Terms: Algorithms

Additional Key Words and Phrases: Wireless networks, data gathering, approximation algorithms, local algorithms

Preliminary versions of this paper appeared in Proc. 25th Symp. on Theoretical Computer Science (2008) and Proc. 4th Workshop on Algorithmic Aspects of Wireless Sensor Networks (2008).

This research was supported by EU COST Action 293 GRAAL, EU ICT-FET 215270 FRONTS, MIUR-FIRB Italy-Israel project RBIN047MH9.

Authors' addresses: V. Bonifaci (corresponding author), Max Planck Institute for Informatics, Campus E1.4, 66123 Saarbrücken, Germany, email: bonifaci@mpi-inf.mpg.de; P. Korteweg, TU Eindhoven, The Netherlands; A. Marchetti-Spaccamela, Sapienza University of Rome, Italy; L. Stougie, VU University and CWI Amsterdam, the Netherlands.

1. INTRODUCTION

Wireless networks are used in many areas of practical interest, such as mobile phone communication, ad-hoc networks, and radio broadcasting. Recent advances in miniaturization of computing devices equipped with short range radios have given rise to strong interest in sensor networks for their relevance in many practical scenarios (environment control, accident monitoring, etc.) [Akyildiz et al. 2002; Pahlavan and Levesque 1995].

Various communication tasks can be distinguished in a wireless network. We focus here on *data gathering*. In many applications of wireless networks, data gathering is a critical operation for extracting useful information from the operating environment: information collected from multiple nodes in the network should be transmitted to a sink that may process the data, or act as a gateway to other networks. We remark that, in the case of wireless sensor networks, sensor nodes have limited computation capabilities, implying that data gathering is an even more crucial operation. Indeed, gathering has received significant attention over the last few years; we cite just a few contributions [Anil Kumar et al. 2004; Florens et al. 2004]. The problem finds also applications in Wi-Fi networks, when many users need to access a gateway using multi-hop wireless relay-routing [Bermond et al. 2006].

In this paper we focus on the problem of designing *simple algorithms* for data gathering with *good approximation guarantees* in *realistic scenarios*. In order to formally assess the performance of the proposed algorithms we focus on objectives that depend on the flow times of the packets. The flow time of a data packet is the time elapsed between its release at the release node and its arrival at the sink. Almost all of the previous literature considered the objective of minimizing the maximum completion time (or *makespan*), see for example [Bar-Yehuda et al. 1992; Bar-Yehuda et al. 1993; Bermond et al. 2006; Florens et al. 2004; Gargano and Rescigno 2006; Anil Kumar et al. 2004; Pelc 2002; Ramanathan and Lloyd 1993] where this objective is applied to a number of different communication tasks. However, flow time is a widely used criterion in scheduling theory that more suitably allows to assess the quality of service when multiple requests occur over time [Chan et al. 2006; Chekuri et al. 2004; Kalyanasundaram and Pruhs 2000; Torng and McCullough 2008], since the end users of a system are typically interested in the response time to their own requests, rather than in the makespan.

The problem of modelling realistic scenarios of wireless sensor networks is complicated by the many parameters that define the communication among nodes and influence the performance of transmissions (see for example [Akyildiz et al. 2002; Schmid and Wattenhofer 2006]). In this paper we assume that stations have a common clock, so that time can be divided into rounds. Each node is equipped with a half-duplex interface; as a result it cannot send and receive during the same round. Typically, not all nodes in the network can communicate with each other directly, hence packets have to be sent through several nodes before they can be gathered at the sink; this is called *multi-hop* routing.

The key issue in our setting is *interference*. A radio signal has a *transmission* radius, the distance over which the signal is strong enough to send data, and an *interference* radius, the distance over which the radio signal is strong enough to interfere with other radio signals. If station i is transmitting data to station j we have *interference* (or *collision*) in communication if j also receives signals from other stations within the interference radius at the same time. There are several proposals for modelling interferences. The simplest possibility is when the interference radius equals the transmission radius: the wireless network is given by a graph, where an edge between nodes i and j represents the fact that stations i and j are within the transmission range of each other, and there is a collision at a node v if two neighbors of v transmit at the same time. This model has been extensively studied in the algorithmic literature (see for example [Bar-Yehuda et al. 1992; Bar-Yehuda et al. 1993; Gargano and Rescigno 2006; Pelc 2002; Ramanathan and Lloyd 1993; Schmid and Wattenhofer 2006] and references therein). In this paper, similarly to [Bermond et al. 2006; Florens et al. 2004], we extend this model via an additional parameter d_I. The integer d_I models the interference radius: a node j successfully receives a data packet if one of its neighbors is transmitting, and no other node within distance d_I from j in the graph is transmitting in the same round.

1.0.1 *The Wireless Gathering Problem.* An instance of the *Wireless Gathering Problem* (Wgp) is given by a static wireless network which consists of several stations (nodes) and one base station (the sink), modeled as a graph, together with the interference radius d_I; over time data packets arrive at stations and have to be gathered at the sink. A feasible solution of an instance of Wgp is a schedule without interference which determines for each packet both the route and times at which it is sent. We consider two different objectives related to the *flow time* of packets, defined as the difference between the arrival time at the sink of a packet, and its release time. One is minimization of the *maximum flow time* over all packets. The other is minimization of the sum of flow times of all packets, also called *total flow time* and equivalent to minimization of average flow time. We call these two problems Fmax-Wgp and Fsum-Wgp, respectively. On the way we will derive some results for the objective of minimizing the *makespan*, i.e., the time needed to gather all the packets, and for the objective of minimizing the *total completion time*, i.e., the sum of the arrival times of the packets at the sink.

1.0.2 *Related work.* There is an extensive literature on flow time minimization in the area of machine scheduling, both for on-line and off-line problems, for which we refer to [Leung 2004]. Here we remark that the two objectives of minimizing maximum flow time and minimizing total flow time lead to fundamentally different problem features, and that results for one problem do not carry over to the other one. In general, minimizing the total flow time appears to be a more difficult problem than minimizing the maximum flow time.

In fact, if we consider on-line algorithms and if the objective function requires to minimize the maximum flow time then the *First In First Out (FIFO)* heuristic is the natural choice: at each time FIFO schedules the earliest released jobs among the unfinished jobs. In the case of uniprocessor scheduling FIFO produces an optimal solution while in the case of parallel machines it gives a $3 - 2/m$ approximation (where m denotes the number of used machines) [Bender et al. 1998]. When the objective function is total flow time the natural heuristic to be used is *Shortest Remaining Processing Time (SRPT)* rule, the on-line strategy that schedules first jobs closer to completion. This heuristic is optimal in the case of one machine but it does not give a constant approximation in the case of parallel machines. In fact, SRPT gives a solution that is $\Theta(\min(\log \frac{n}{m}, \log P))$ approximate, where n and m denote respectively the number of jobs and the number of machines and P denotes the ratio between the longest and the smallest processing time [Leonardi and Raz 2007]. In the same paper it is shown that no on-line randomized algorithm can achieve a better competitive ratio.

The literature on communication protocols for radio networks is very extensive so in the following we will limit ourselves to cite the results that are more relevant to our work. The Wireless Gathering Problem was studied by Bermond et al. [2006] in the context of wireless access to the Internet in villages. The authors proved that the problem of minimizing the makespan is **NP**-hard and presented a greedy algorithm with asymptotic approximation ratio at most 4. In previous work [Bonifaci et al. 2008] we considered arbitrary release times and proposed an on-line greedy algorithm with the same approximation ratio.

Bar-Yehuda et al. [1993] considered distributed algorithms for WGP when the objective is the minimization of the makespan. Their model is a special case of our model, with $d_I = 1$ and no release dates.

Kumar et al. [2005] give an overview of other interference models, including the so-called *distance-2 interference model*. The distance-2 interference model is similar to our interference model with $d_I = 1$, with the added constraint that no two transmitting nodes should be adjacent; we observe that this requirement might pose an unnecessary restriction.

Kumar et al. [2004] studied the more general *end-to-end* transmission problem, where each of the packets may have a different source *and* destination in the network. Under the assumption of the above mentioned distance-2 interference model, minimization of the maximum completion time of the makespan is considered, and hardness results and approximation algorithms for arbitrary graphs and disk graphs are presented. The authors present distributed algorithms for packet scheduling over fixed routing paths, and use a linear program in order to determine the paths. By contrast, we use a shortest paths tree to fix the routing paths, which is easier to implement in a distributed setting.

Florens et al. [2004] considered the minimization of the completion time of data gathering in a setting with unidirectional antennas. They provided a 2-approximation algorithm for tree networks and an optimal algorithm for line networks. Gargano and Rescigno [2006] gave a polynomial time algorithm for the special case of the same model in which each node has exactly one packet to send.

Finally, we observe that many papers study broadcasting in wireless networks [Bar-Yehuda et al. 1992; Pelc 2002]. However, we stress that data gathering and broadcasting are substantially different tasks in the context of packet networks. In particular, the idea of reversing a broadcast schedule to obtain a gathering schedule could only work if data can be aggregated. In general it does not and hence results do not carry over.

1.0.3 *Results.* In the present paper we focus on simple local on-line algorithms for FMAX-WGP and FSUM-WGP. Our algorithms are neither centralized nor fully distributed, but *local*, in the sense that they can be executed using local information only (see also [Linial 1992]). In particular, it suffices that a node is informed about the state of nodes within distance $d_I + 1$ from it. On the other hand, our lower bounds hold for centralized algorithms as well. Our algorithms are also *on-line*. At time t an on-line algorithm makes its decisions on events that occur at or before t, ignoring information about future events. Competitive analysis compares the solution of the on-line algorithm with the optimal solution obtained by an omniscient adversary who provides the input instance. We refer the reader to [Borodin and El-Yaniv 1998] for a comprehensive survey on on-line algorithms and competitive analysis. We restrict to simple local algorithms because they may be amenable for implementation or faithfully represent algorithms used in practice. In fact, we think that algorithms that are too sophisticated are often impractical and have mainly theoretical interest.

We will give a mathematical formulation of the problems in Section 2. In particular we will see that they can be modeled as clean combinatorial optimization problems.

In Section 3 we give inapproximability results for FMAX-WGP and FSUM-WGP. We prove that FMAX-WGP and FSUM-WGP on m packets cannot be approximated in polynomial time within $\Omega(m^{1-\epsilon})$, for any $\epsilon \in (0, 1)$, unless $\mathbf{P} = \mathbf{NP}$.

In Sections 4 and 5 we present two on-line polynomial time algorithms for FMAX-WGP and FSUM-WGP, respectively. For FMAX-WGP the algorithm is based on the FIFO rule, whereas for FSUM-WGP it is based on the SRPT rule. Since neither of the two problems allows for a reasonable approximation ratio, we assess the performance of the algorithms using resource augmentation.

Resource augmentation was introduced in the context of machine scheduling in [Kalyanasundaram and Pruhs 2000]: the idea is to study the performance of on-line algorithms which are given processors faster than the adversary. Intuitively, this has been done to compensate an on-line scheduler for its lack of future information. Such an approach has led to a number of interesting results showing that moderately faster processors are sufficient to attain satisfactory performance guarantee for various scheduling problems, see for example [Chekuri et al. 2004; Kalyanasundaram and Pruhs 2000].

Surprisingly, in the case of FMAX-WGP and FSUM-WGP a modest resource augmentation allows to compensate not only the lack of future information but also the approximation hardness of the problem. Namely, we show that the algorithms attain an objective value not larger than that of an optimal solution, assuming that they are run at speed 5 times faster than the optimal solution.

2. MATHEMATICAL PRELIMINARIES

We formulate WGP as a graph optimization problem. The model can be seen as a generalization of a well-studied model for packet radio networks [Bar-Yehuda et al. 1992; Bar-Yehuda et al. 1993]. It has also been used in more recent work [Bermond et al. 2006; Florens et al. 2004]. We summarize it for independent reading.

An instance of WGP consists of an unweighted, undirected graph $G = (V, E)$, a *sink* node $s \in V$, a positive integer d_I, and a set of *data packets* $J = \{1, 2, \ldots, m\}$. Each packet $j \in J$ has an *origin* $o_j \in V$ and a *release date* $r_j \in \mathbb{Z}_+$.

We assume that time is discrete; we call a time unit a *round*. The rounds are numbered $0, 1, 2, \ldots$. During each round a node can be in one of three states: *sending* a packet, *receiving* a packet, or *inactive*. Node u can send a packet to node v during a round, only if u and v are adjacent. If u sends a packet j to v in some round, then the pair (u, v) is said to be a *call* from u to v. For each pair $u, v \in V$, the *distance* between u and v, denoted by $d(u, v)$, is the length of a shortest path from u to v in G. Two calls (u, v) and (u', v') *interfere* if they occur in the same round and either $d(u', v) \leq d_I$ or $d(u, v') \leq d_I$; otherwise the calls are *compatible*. For this reason, the parameter d_I is called the *interference radius*.

For every packet $j \in J$, the release date r_j specifies the time at which the packet enters the network, i.e. packet j cannot be sent before round r_j. In the off-line version the entire instance is completely known at time 0; in the on-line version information about a packet becomes known only at its release date.

A solution for a WGP instance is a schedule of compatible calls such that all packets are ultimately sent to the sink. Notice that while in principle each radio transmission could broadcast the same packet to multiple destinations, in the gathering problem having more than one copy of the same packet does not help, as it suffices to keep the one that will arrive first at the sink. Thus, we assume that at any time there is a unique copy of each packet. Also, in the model we consider, packets cannot be aggregated.

Given a schedule, let x_j^t be the unique node holding packet j at time t. The integer $C_j := \min\{t : x_j^t = s\}$ is called the *completion time* of packet j, while $F_j := C_j - r_j$ is the *flow time* of packet j. In this paper we are interested in the minimization of $\max_j F_j$ (FMAX-WGP) and $\sum_j F_j$ (FSUM-WGP). As an intermediate step or as a byproduct of the analysis of these two problems, we also consider the minimization of $\max_j C_j$ (CMAX-WGP) and of $\sum_j C_j$ (CSUM-WGP).

Some additional notation: we denote by $\delta_j := d(o_j, s)$ the minimum number of calls required for packet j to reach s. We also define $\gamma := d_I + 2$, which, roughly stated, is an upper bound on the number of rounds during which a packet needs to be forwarded before a new packet can be forwarded safely from the same origin over the same path. The *critical region* is the set $\{v \in V \mid d(s, v) \leq \lfloor (d_I - 1)/2 \rfloor\}$, which is a region around s in which no two nodes can receive a message in the same round. Related to this region we define $\gamma^* := \lfloor (d_I + 1)/2 \rfloor$, which is then a lower bound, because of interference, on the average inter arrival time at s between any two messages that are released outside the critical region.

We analyze the performance of our algorithms using the standard worst case analysis techniques of approximation ratio analysis, as well as resource augmentation. Given a WGP instance $\mathcal{I}$ and an algorithm ALG, we define $\mathcal{C}(\mathcal{I})$ as the cost of ALG and $\mathcal{C}^*(\mathcal{I})$ as the cost of the optimal solution on $\mathcal{I}$. A polynomial-time algorithm is called an α-approximation if for any instance $\mathcal{I}$ we have $\mathcal{C}(\mathcal{I}) \leq \alpha \cdot \mathcal{C}^*(\mathcal{I})$.

In the resource augmentation paradigm, the algorithm is allowed to use more resources than the adversary. We consider augmentation based on speed, meaning that the algorithm can schedule compatible calls with higher speed than an optimal algorithm. For any $\sigma \geq 1$, we call an algorithm a σ-speed algorithm if the time used by the algorithm to schedule a set of compatible calls is $1/\sigma$ time units. Thus, the ith round occurs during time interval $[i/\sigma, (i + 1)/\sigma)$. Notice that the input and, in particular, the release dates of packets, are not affected by the value of σ.

We finally call a σ-speed algorithm *optimal* if its cost is bounded above by the cost of an optimal unit-speed solution.

3. INAPPROXIMABILITY

In this section we prove inapproximability results for FMAX-WGP and FSUM-WGP. To prove these results we consider the so-called *induced matching* problem. A matching M in a graph G is an *induced matching* if no two edges in M are joined by an edge of G [1].

INDUCED BIPARTITE MATCHING (IBM)
Instance: a bipartite graph G and an integer k.
Question: does G have an induced matching of size at least k?

The following rather straightforward relation between compatible calls in a bipartite graph and induced matchings will be crucial in the following.

PROPOSITION 3.1. *Let $G = (U, V, E)$ be a bipartite graph with node sets (U, V) and edge set E. Then, a set $M \subseteq E$ is an induced matching if and only if the calls corresponding to edges of M, directed from U to V, are all pairwise compatible, assuming $d_I = 1$.* $\square$

A *grid graph* is a graph whose nodes can be represented as points of the integer grid $\mathbb{Z}^2$ so that two nodes are connected by an edge if and only if the Euclidean distance between the corresponding points is at most 1. Notice that grid graphs are always bipartite.

LEMMA 3.2. INDUCED BIPARTITE MATCHING *is* NP*-hard even when restricted to grid graphs.*

PROOF. Recall that a *dominating set* of a graph $G = (V, E)$ is a set of nodes $S \subseteq V$ such that every node in $V \setminus S$ is adjacent to some node in S. Given a graph G and an integer k, the DOMINATING SET problem consists in determining whether G has a dominating set of size at most k. We will use the fact that DOMINATING SET is **NP**-hard for grid graphs [Clark et al. 1990]. Let $\mathbf{mds}(G)$ and $\mathbf{mim}(G)$ denote, respectively, the size of a minimum dominating set and of a maximum induced matching of a graph G. Moreover, let $S(G)$ denote the graph obtained from G by subdividing every edge in two; that is, each edge of G is replaced by a path of two edges. Then Ko and Shepherd [2003] observe (without proof) that for any graph G, $\mathbf{mds}(G) + \mathbf{mim}(S(G)) = n$ where n is the number of nodes of G. From this relationship it follows that if DOMINATING SET is **NP**-hard for a class of graphs $\mathcal{G}$, then INDUCED BIPARTITE MATCHING is **NP**-hard for the class $\{S(G) : G \in \mathcal{G}\}$. This implies that INDUCED BIPARTITE MATCHING is **NP**-hard for grid graphs, because $S(G)$ is a grid graph whenever G is a grid graph.

For completeness, we also include a proof of Ko and Shepherd's assertion. Let $D \subseteq V(G)$ be a minimum dominating set for G and let $D' = V(G) \setminus D$. Since D is a dominating set, every node $v \in D'$ has at least a neighbor in D; call it $\mathrm{dom}(v)$. Since $(v, \mathrm{dom}(v))$ is an edge in G, there is a corresponding node in $S(G)$; call this node $e(v)$. We now claim that the edge set $M := \{(v, e(v)) : v \in D'\}$ is an

[1]Induced matchings are also known as *distance-2 matchings* [Anil Kumar et al. 2004].

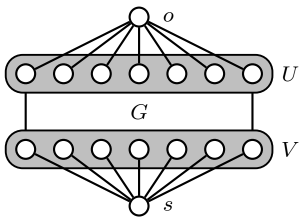

Fig. 1. The network N constructed in the proof of Theorem 3.3

induced matching in $S(G)$. If this is not the case, then there exist $v_1, v_2 \in D'$ such that $(v_1, e(v_1)) \in M$, $(v_2, e(v_2)) \in M$, and $(v_1, e(v_2))$ is an edge of $S(G)$. But this implies that $\mathrm{dom}(v_2) = v_1$, contradicting the assumption that $v_1 \in D'$. This shows that $\mathbf{mim}(S(G)) \geq |M| = n - |D| = n - \mathbf{mds}(G)$.

In the other direction, let M be a maximum induced matching in $S(G)$. Let D be the set of nodes of G to which M is not incident; so $|D| = n - |M|$. Let v be any node not in D; since M is incident to v, v has one mate in M, call it $e'(v)$, that has degree 2 in $S(G)$. Then the other neighbor of $e'(v)$ in $S(G)$ (other than v) cannot have an edge of M incident to it and thus must belong to D. This shows that D must be a dominating set and so $\mathbf{mds}(G) \leq |D| = n - |M| = n - \mathbf{mim}(G)$. □

THEOREM 3.3. *Unless* $\mathbf{P} = \mathbf{NP}$, *no polynomial-time algorithm can approximate* FMAX-WGP *or* FSUM-WGP *within a ratio better than* $\Omega(m^{1-\epsilon})$ *for any* $\epsilon \in (0, 1)$.

PROOF. Let (G, k) be an instance of IBM, with $G = (U, V, E)$ a grid graph. Without loss of generality we assume $k > 1$. We construct a 4-layer network N with a unique source o (first layer), a clique on U and a clique on V (middle layers), and a sink s (last layer). Source o is adjacent to each node in U, and s to each node in V. The edges between U and V are the same as in G (see Figure 1). We set $d_I = 1$.

The WGP instance consists of m packets with origin o, where m will be specified later. They are divided into m/k groups of size k. Each packet in the h-th group has release date $(k+1)h$, $h = 0, \ldots, m/k - 1$. Rounds $(k+1)h$ till $(k+1)(h+1) - 1$ together are a *phase*.

We prove that if G has an induced matching of size k, there is a gathering schedule in N such that every message has flow time at most $2k+1$, while if G has no induced matching of size more than $k - 1$, then every schedule will have large maximum and total flow time.

Assume G has an induced matching M of size k, say (u_i, v_i), $i = 0 \ldots k-1$. Then consider the following gathering schedule. In each phase, the k new packets at o are transmitted, necessarily one by one, to layer U while old packets at layer V (if any) are absorbed at the sink; then, in a *single* round, the k new packets move from U to V via the matching edges. More precisely, each phase can be scheduled in $k + 1$ rounds as follows:

1. for $i = 0, \ldots, k-1$ execute in the ith round the two calls (o, u_i) and $(v_{i+1 \bmod k}, s)$;
2. in the kth round, execute simultaneously all the calls (u_i, v_i), $i = 0, \ldots, k - 1$.

The maximum flow time of the schedule is $2k + 1$, as a packet released in phase h reaches the sink before the end of phase $h + 1$. This implies that the total flow time of this schedule is bounded by $(2k + 1)m$.

In the other direction, assume that each induced matching of G is of size at most $k-1$. By Proposition 3.1, at most $k-1$ calls can be scheduled in any round from layer U to layer V. We ignore potential interference between calls from o to U and calls from V to s; doing so may only decrease the cost of a schedule. As a consequence, we can assume that each packet follows a shortest path from o to s. Notice however that, due to the cliques on the layers U and V, no call from U to V is compatible with a call from o to U, or with a call from V to s.

Let m_o and m_U be the number of packets at o and U, respectively, at the beginning of a given phase. We associate to the phase a potential value $\psi := km_o + m_U$, and we show that at the end of the phase the potential must have increased by at least one unit. Let c_o and c_U denote the number of calls from o to U and from U to V, respectively, during the phase. Since a phase consists of $k+1$ rounds, and in each round at most $k-1$ calls are scheduled from U to V, we have $c_o + c_U/(k-1) \leq k+1$, or, equivalently,

$$(k-1)c_o + c_U \leq (k-1)(k+1). \tag{1}$$

If m'_o, m'_U are the number of packets at o and U at the beginning of the next phase, and $\psi' = km'_o + m'_U$ is the new potential, we have

$$\begin{aligned} m'_o &= m_o + k - c_o \\ m'_U &= m_U + c_o - c_U \\ \psi' - \psi &= k(m'_o - m_o) + m'_U - m_U \\ &= k(k - c_o) + c_o - c_U \\ &= k^2 - (k-1)c_o - c_U \\ &\geq k^2 - (k-1)(k+1) \\ &= 1 \end{aligned}$$

where the inequality uses (1).

Thus, consider the situation after $\Psi := m/k$ phases. The potential has become at least Ψ. By definition of the potential, this implies that at least $\Psi/k = m/k^2$ packets reside at either o or U; in particular, they have been released but not yet absorbed at the sink. Since the sink cannot receive more than one packet per round, this clearly implies a maximum flow time of at least m/k^2, and a total flow time of at least $m^2/4k^4$.

In the case of FMAX-WGP, we now set $m = \Theta(k^{\frac{3}{\epsilon}})$. The maximum flow time of any schedule is thus at least $m/k^2 = \Theta(m^{1-\frac{2}{3}\epsilon})$. Comparison to the $2k+1$ bound in case there exists an induced matching of size k implies now the inapproximability bound of $\Omega(m^{1-\epsilon})$.

Similarly, in the case of FSUM-WGP we can set $m = \Theta(k^{\frac{5}{\epsilon}})$. The total flow time of any schedule is in this case at least $m^2/4k^4 = \Theta(m^{2-\frac{4}{5}\epsilon})$ and comparison to the $(2k+1)m$ bound in case there exists an induced matching of size k implies the inapproximability bound of $\Omega(m^{1-\epsilon})$. □

In cases where the packets are routed via shortest paths to the sink – a behavior common to many gathering protocols – the result of Theorem 3.3 can be strengthened further to an $\Omega(m)$ lower bound, the proof of which we omit (see [Bonifaci et al. 2008]).

The reader might wonder if the inapproximability result of Theorem 3.3 also holds when the underlying network has to satisfy geometric constraints. The answer is yes, as we argue in the following.

A (*3-dimensional*) *unit ball graph* is a graph whose nodes can be represented as points in $\mathbb{R}^3$ so that two nodes are connected by an edge if and only if the Euclidean distance between the corresponding points is at most 1. A *quasi-unit disk graph* with *threshold* $\rho \in (0,1]$ is a graph whose nodes can be represented as points in $\mathbb{R}^2$ so that two nodes are connected by an edge whenever the distance between the corresponding points is at most ρ, and are not connected by an edge whenever the distance is larger than 1; when the distance is between ρ and 1, the nodes may or may not be adjacent. Both unit ball graphs and quasi-unit disk graphs are well-studied models for wireless networks [Schmid and Wattenhofer 2006].

THEOREM 3.4. *Theorem 3.3 holds even when* FMAX-WGP *or* FSUM-WGP *are restricted to unit ball graphs.*

PROOF. We will show that the graph N which is part of the input for FMAX-WGP or FSUM-WGP (the one depicted in Figure 1) is a unit ball graph; that is, it can be embedded into $\mathbb{R}^3$ with the Euclidean metric. To this end, let $G = (U, V, E)$ be the (bipartite) grid graph that is the input to the reduction. Recall that the network N constructed in the proof of Theorem 3.3 (Figure 1) consists of a copy of G plus two nodes – the source o and the sink s – as well as the edges that connect $\{o\} \cup U$ as a clique and $\{s\} \cup V$ as clique.

Let $f_x, f_y : U \cup V \to \mathbb{Z}$ be the embedding of the grid graph G into $\mathbb{Z}^2$. Notice that because G is a grid graph, the bipartition of G can be chosen such that $f_x(w)+f_y(w)$ is odd when $w \in U$ and even when $w \in V$. Also, let

$$\Delta := \max_{w,w' \in U \cup V} \sqrt{(f_x(w) - f_x(w'))^2 + (f_y(w) - f_y(w'))^2},$$

or in other words, let Δ be the maximum Euclidean distance between two nodes in the bidimensional embedding of G.

In the following we assume that two nodes are connected if and only if they are at most at distance $\sqrt{\Delta^2 + 1}$, instead of unit distance, in the embedding; this is clearly equivalent, as we can always scale coordinates. We use the following embedding:

—any $u \in U$ is mapped to $(f_x(u), f_y(u), \Delta)$;

—any $v \in V$ is mapped to $(f_x(v), f_y(v), 0)$;

—the source o is mapped to $(f_x(v), f_y(v), \Delta + 1)$, where v is an arbitrarily chosen node of $U \cup V$;

—the sink s is mapped to $(f_x(v), f_y(v), -1)$, where v is an arbitrarily chosen node of $U \cup V$.

See Figure 2 for an illustration of the embedding (the edges of the cliques on U and V have not been drawn). Observe that any two nodes that are adjacent in G are at distance exactly $\sqrt{\Delta^2 + 1}$ in the embedding. On the other hand, nodes on the same side of the bipartition of G are at distance at most Δ in the embedding, and so they form a clique. We leave it to the reader to verify that the resulting edges are in fact exactly the edges of N. $\square$

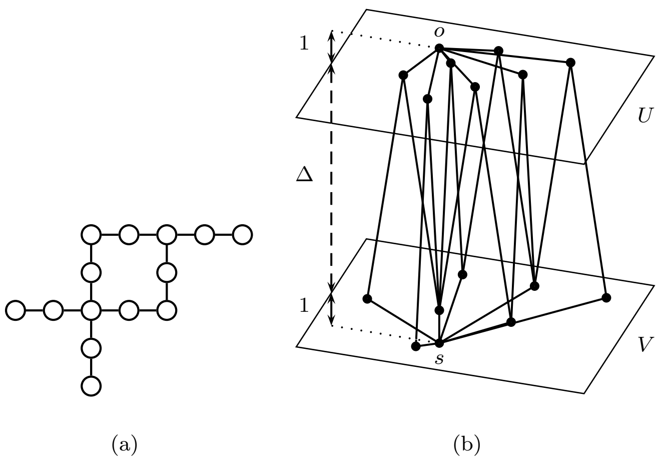

Fig. 2. (a) a grid graph G; (b) the three-dimensional embedding of the resulting network N as a unit ball graph (the cliques on U and V are not shown)

Theorem 3.5. *Theorem 3.3 holds even when* Fmax-Wgp *or* Fsum-Wgp *are restricted to quasi-unit disk graphs (for any threshold $\rho < 1$).*

Proof. We have to show that the graph N can be represented as a quasi-unit disk graph for any $\rho < 1$. The embedding of N in $\mathbb{R}^2$ is simple: we map any node in $U \cup \{o\}$ to point $(0, 0)$ and any node in $V \cup \{s\}$ to $((\rho + 1)/2, 0)$. All the nodes in $U \cup \{o\}$ will then form a clique, as they need to; similarly for nodes in $V \cup \{s\}$. Because $\rho < (\rho + 1)/2 < 1$, edges between $U \cup \{o\}$ and $V \cup \{s\}$ are arbitrary, and so the corresponding edges of N can certainly be represented. □

An even more restrictive model for wireless networks is that of *unit disk graphs*, which is the restriction of unit ball graphs to points in $\mathbb{R}^2$, or, equivalently, the restriction of quasi-unit disk graphs to the case where the threshold ρ is exactly 1. We leave open the problem of proving hardness results for Fmax-Wgp and Fsum-Wgp on this class of graphs.

4. APPROXIMATION ALGORITHMS FOR MAXIMUM FLOW TIME

In this section we present and analyze a FIFO algorithm for Wgp. First, we show that FIFO is a 5-approximation for Cmax-Wgp. Note that the best approximation algorithm known for Cmax-Wgp is 4-approximate [Bonifaci et al. 2008]; the main interest in analyzing FIFO is that we use it as a subroutine in an algorithm for Fmax-Wgp which uses resource augmentation. Next, we prove that this algorithm with resource augmentation is a σ-speed optimal algorithm, for any $\sigma \geq 5$, for both Cmax-Wgp and Fmax-Wgp.

4.1 An approximation algorithm for Cmax-Wgp

In this section we extend the results from [Bonifaci et al. 2008] to show that a FIFO-type algorithm achieves a constant approximation for Cmax-Wgp. This fact, together with the specific properties of the algorithm, will then be used in Section 4.2 to derive bounds for the minimization of maximum flow time.

The algorithm can be seen a special case of a general scheme for which it is possible to prove an upper bound on the completion time [Bonifaci et al. 2008]. In this scheme, called PRIORITY GREEDY, each packet is assigned a unique priority based on some algorithm-specific rules. Then, in each round, packets are considered in order of decreasing priority and are sent towards the sink as long as there is no interference with higher priority packets (Algorithm 1).

Algorithm 1 PRIORITY GREEDY

```
Let $1, \dots, m$ be the packets in order of decreasing priority
for $t = 0, 1, 2, \dots$ do
  for $j = 1$ to $m$ do
    In round $t$, send $j$ (if available) to the next hop along an arbitrary shortest
    path from $x_j^t$ to the sink, unless this creates interference with a packet $j'$
    with $j' < j$
  end for
end for
```

We first derive upper bounds on the completion time C_j of each packet j in a PRIORITY GREEDY solution.

We say that packet j is *blocked* in round t if $t \geq r_j$ but j is not sent in round t. Note that in a PRIORITY GREEDY algorithm a packet can only be blocked due to interference with a higher priority packet. We define the following *blocking relation* on the packets of a PRIORITY GREEDY schedule: $k \prec j$ if, in the last round t in which j is blocked, k is a packet that is sent in that round, satisfies $d(x_j^t, x_k^t) \leq d_I + 1$, and has a priority higher than j. Such a k must exist as long as j has been blocked at least once. In case there are multiple candidates for k, we break ties arbitrarily, so that there exists at most one k such that $k \prec j$.

The blocking relation induces a directed graph $F = (J, A)$ on the packet set J with an arc (k, j) for each $k, j \in J$ such that $k \prec j$. Observe that, for any PRIORITY GREEDY schedule, F is a directed forest and the root of each tree of F is a packet which is never blocked. For each j let $T(j) \subseteq F$ be the tree of F containing j, $b(j) \in J$ be the root of $T(j)$, and $P(j)$ the set of packets along the path in F from $b(j)$ to j. Finally, define $\pi_j := \min\{\delta_j, \gamma^*\}$ and $R_j := r_j + \delta_j - \pi_j$ (recall from Section 2 that δ_j is the initial distance of packet j to the sink, while $\gamma^* = \lfloor (d_I + 1)/2 \rfloor$ gives the minimum number of rounds during which a packet initially released outside the critical region will necessarily impede the transmission of any other packet inside the region).

We make use of an upper bound on the completion time of each packet and a lower bound on the maximum completion time of subsets of packets, both of which were derived for the first time in [Bonifaci et al. 2008]. We include short proofs for the sake of completeness.

The intuition behind the upper bound is that packets can only interfere with each other if they are within distance $d_I + 1$, so if a packet is blocked, its distance to the sink is almost the same as the blocking packet. Since the blocking packet has a higher priority, if we already know by induction that it has small completion time – and thus short distance to the sink at the time the blocking occurred – then the completion time of the blocked packet cannot be much larger.

LEMMA 4.1 [BONIFACI ET AL. 2008]. *For each packet* $j \in J$, $C_j \leq R_{b(j)} + \frac{\gamma}{\gamma^*}\sum_{i \in P(j)} \pi_i$.

PROOF. Consider any packet j. The proof is by induction on the height of $T(j)$. If the height of $T(j)$ is zero, then j is never blocked, $b(j)$ equals j, and the lemma is easily seen to hold. Otherwise, let t be the last round during which j is blocked by some packet k, $k \prec j$, so that $d(x_j^t, x_k^t) \leq d_I + 1$. Observe that $d(x_k^t, s) \leq C_k - t$, otherwise k does not reach the sink by time C_k. From round $t+1$ on, j traverses an edge each round, reaching the sink at time

$$\begin{aligned} C_j &\leq t + 1 + d(x_j^t, x_k^t) + d(x_k^t, s) \\ &\leq t + 1 + d_I + 1 + C_k - t \\ &= C_k + \gamma. \end{aligned}$$

In addition, $C_j \leq C_k + \delta_j$, since j is never at distance larger than δ_j from the sink, and is never blocked from time C_k on. Thus $C_j \leq C_k + \min\{\delta_j, \gamma\} \leq C_k + (\gamma/\gamma^*)\pi_j$ and the lemma follows by applying the induction hypothesis to C_k. □

The lower bound follows easily from the fact that the critical region cannot be used simultaneously for different transmissions.

LEMMA 4.2 [BONIFACI ET AL. 2008]. *Let* $S \subseteq J$ *be a nonempty set of packets, and let* C_i^* *denote the completion time of packet* i *in some feasible schedule. Then there is* $k \in S$ *such that* $\max_{i \in S} C_i^* \geq R_k + \sum_{i \in S} \pi_i$.

PROOF. Any feasible solution to WGP gives a feasible solution to a preemptive single machine scheduling problem, in which the release time of job j (corresponding to packet j) is R_j and its processing time is π_j. Ignoring interference outside the critical region can only decrease the cost of a solution. The lemma now follows by noticing that, in the scheduling problem, the makespan must be at least the release date of some job k, plus the sum of the processing times of all the jobs. □

The algorithm we analyze is a special case of the PRIORITY GREEDY scheme, in which higher priority is given to packets with earlier release dates (ties broken arbitrarily). In other words, we analyze Algorithm 1 under the assumption that j has higher priority than k if and only if either $r_j < r_k$, or $r_j = r_k$ and $j < k$. We call the resulting algorithm FIFO after the well-known *First In, First Out* policy in scheduling and service systems, though in our case packets do not necessarily arrive at the sink in order of their priority.

THEOREM 4.3. FIFO *is a* $(1 + \frac{\gamma}{\gamma^*})$*-approximation algorithm for* CMAX-WGP.

PROOF. Let j be the packet having maximum C_j, and consider $T(j)$, the tree containing j in the forest induced by the blocking relation. With some abuse of notation, in the following we also regard $T(j)$ as a set of packets (the nodes of

$T(j)$). We can now apply Lemma 4.2 with $S = T(j)$ to obtain

$$\max_{i\in T(j)} C_i^* \geq r_k + \delta_k + \sum_{\substack{i\in T(j)\\ i\neq k}} \pi_i \tag{2}$$

where k is some packet in $T(j)$. On the other hand, by using Lemma 4.1,

$$\begin{aligned} C_j &\leq R_{b(j)} + \frac{\gamma}{\gamma^*}\sum_{i\in P(j)} \pi_i \\ &= r_{b(j)} + \delta_{b(j)} - \pi_{b(j)} + \frac{\gamma}{\gamma^*}\sum_{i\in P(j)} \pi_i \\ &= r_{b(j)} + \frac{\gamma}{\gamma^*}\min\{\delta_k, \gamma^*\} + \frac{\gamma}{\gamma^*}\sum_{\substack{i\in P(j)\\ i\neq k}} \pi_i + \delta_{b(j)} - \pi_{b(j)} \\ &\leq \frac{\gamma}{\gamma^*}\left(r_k + \delta_k + \sum_{\substack{i\in T(j)\\ i\neq k}} \pi_i\right) + \delta_{b(j)}. \end{aligned} \tag{3}$$

where we used the fact that, by definition of FIFO, we have $r_{b(j)} \leq r_k$. Equations (2) and (3), together with the observation that $\max_{i\in T(j)} C_i^* \geq \delta_{b(j)}$, prove the theorem. □

It is easy to verify that $2 \leq \gamma/\gamma^* \leq 4$ for all d_I, and that $\gamma/\gamma^* = 3$ when $d_I = 1$.

Corollary 4.4. FIFO *is a* 5*-approximation algorithm for* Cmax-Wgp*. When* $d_I = 1$*,* FIFO *is a* 4*-approximation for* Cmax-Wgp*.*

The bound on the approximation ratio of FIFO is slightly worse than that of a Priority Greedy algorithm where priorities are based on R_j, which is a γ/γ^*-approximation (see [Korteweg 2008, Theorem 5.7, p.77] for a construction showing that FIFO is strictly worse than a γ/γ^*-approximation). As mentioned before, the interest of this result is its importance in proving good bounds for the minimization of the maximum flow time, where we will use FIFO as a subroutine of our algorithm.

4.2 A resource augmentation bound for Fmax-Wgp

Motivated by the hardness result of Section 3, we study algorithms under resource augmentation. In this context we study σ-speed algorithms, in which data packets are sent at a speed that is σ times faster than the solution we compare them to. In particular, consider the following algorithm (Algorithm 2).

Algorithm 2 σ-FIFO

```
1. Create a new instance $\mathcal{I}'$ by multiplying release dates: $r'_j := \sigma r_j$;
2. Run FIFO on $\mathcal{I}'$;
3. Speed up the schedule thus obtained by a factor of $\sigma$.
```

The schedule constructed by σ-FIFO is a feasible σ-speed solution to the original problem because of step 1. We will show that σ-FIFO is optimal for both Cmax-Wgp and Fmax-Wgp, if $\sigma \geq \gamma/\gamma^* + 1$. The following lemma is crucial.

LEMMA 4.5. *If σ-FIFO is a σ-speed optimal algorithm for* CMAX-WGP*, then it is also a σ-speed optimal algorithm for* FMAX-WGP.

PROOF. Let C_j^* and C_j^σ be the completion time of data packet j in an optimal solution and in a σ-FIFO solution, respectively, and let F_j^* and F_j^σ be the flow time of data packet j in these solutions. Suppose σ-FIFO is a σ-speed optimal algorithm for CMAX-WGP, hence we have $\max_{j\in J} C_j^\sigma \le \max_{j\in J} C_j^*$. We show that this inequality implies, for any time t,

$$\max_{j\in J,\, r_j=t} C_j^\sigma \le \max_{j\in J,\, r_j\le t} C_j^*. \tag{4}$$

We prove inequality (4) by contradiction. Suppose it is false, then there is an instance of minimum size (number of data packets) for which it is false. Also, let t_0 be the first round in such an instance for which it is false. By definition, σ-FIFO schedules each data packet j definitively in round r_j; no data packet is rescheduled strictly after its release time. I.e., the algorithm determines the completion time C_j^σ in round r_j. Thus, if inequality (4) were false, then

$$C_i^\sigma > \max_{j\in J,\, r_j\le t_0} C_j^*, \tag{5}$$

for some data packet i with $r_i = t_0$, and because $\mathcal{I}$ is a minimum size instance the instance does not contain any data packets released after round t_0. But then (5) would contradict $\max_{j\in J} C_j^\sigma \le \max_{j\in J} C_j^*$.

Using (4) we have

$$\begin{aligned}\max_{j\in J} F_j^\sigma &= \max_t \left(\max_{j\in J,\, r_j=t} C_j^\sigma - t\right) \le \max_t \left(\max_{j\in J,\, r_j\le t} C_j^* - t\right)\\ &\le \max_t \left(\max_{j\in J,\, r_j\le t} F_j^*\right) = \max_{j\in J} F_j^*.\end{aligned}$$

□

THEOREM 4.6. *For $\sigma \ge \gamma/\gamma^* + 1$, σ-FIFO is a σ-speed optimal algorithm for both* CMAX-WGP *and* FMAX-WGP.

PROOF. By Lemma 4.5, it suffices to prove that σ-FIFO is a σ-speed optimal algorithm for CMAX-WGP.

Let C_j be the completion time of any data packet j in the σ-FIFO solution on instance $\mathcal{I}$, and let C'_j be the completion time of j in the FIFO solution on the instance $\mathcal{I}'$ (see the algorithm description). By construction $C_j = C'_j/\sigma$. Then the upper bound of Lemma 4.1 implies that $C'_j \le R'_{b(j)} + (\sigma-1)\sum_{i\in P(j)} \pi_i$, where $R'_{b(j)} = \sigma r_{b(j)} + \delta_{b(j)} - \pi_{b(j)}$. Hence,

$$C_j = C'_j/\sigma \le \frac{1}{\sigma} R'_{b(j)} + \sum_{i\in P(j)} \pi_i \;\le\; r_{b(j)} + \frac{1}{\sigma}\delta_{b(j)} + \frac{\sigma-1}{\sigma}\sum_{i\in P(j)} \pi_i. \tag{6}$$

Since in any solution packet $b(j)$ has to reach the sink, we clearly have

$$\max_{i\in P(j)} C_i^* \ge C_{b(j)}^* \ge r_{b(j)} + \delta_{b(j)}. \tag{7}$$

Also, by Lemma 4.2, for some $k \in P(j)$,

$$\max_{i\in P(j)} C_i^* \geq R_k + \sum_{i\in P(j)} \pi_i \geq r_k + \sum_{i\in P(j)} \pi_i \geq r_{b(j)} + \sum_{i\in P(j)} \pi_i, \tag{8}$$

where the last inequality follows from $b(j)$ having lowest release time in $P(j)$, by definition of FIFO. Combining (6), (7) and (8), we obtain

$$\begin{aligned}\max_{i\in P(j)} C_i^* &= \frac{1}{\sigma}\max_{i\in P(j)} C_i^* + \frac{\sigma-1}{\sigma}\max_{i\in P(j)} C_i^* \\ &\geq \frac{1}{\sigma}\left(r_{b(j)} + \delta_{b(j)}\right) + \frac{\sigma-1}{\sigma}\left(r_{b(j)} + \sum_{i\in P(j)} \pi_i\right) \\ &= r_{b(j)} + \frac{1}{\sigma}\delta_{b(j)} + \frac{\sigma-1}{\sigma}\sum_{i\in P(j)} \pi_i \geq C_j.\end{aligned}$$

□

We use again that $2 \leq \gamma/\gamma^* \leq 4$ for all d_I, and that $\gamma/\gamma^* = 3$ when $d_I = 1$ to obtain the main result of this section.

Corollary 4.7. 5-FIFO *is a* 5*-speed optimal algorithm for* Cmax-Wgp *and* Fmax-Wgp*. When* $d_I = 1$*,* 4-FIFO *is a* 4*-speed optimal algorithm for* Cmax-Wgp *and* Fmax-Wgp*.*

4.3 FIFO without resource augmentation

We show here that, modulo some constant factor, FIFO is best possible among algorithms that use shortest paths. As we have seen in Section 3, Fmax-Wgp is extremely hard to approximate without resource augmentation: no bound better than $\Omega(m^{1-\epsilon})$ is possible. Moreover, algorithms that route along shortest paths cannot do better than $\Omega(m)$ (recall the remark after Theorem 3.3).

Theorem 4.8. FIFO *is an* $O(m)$*-approximation for* Fmax-Wgp*.*

Proof. Since every packet must be gathered at the sink, clearly $\max_j F_j^* \geq \max_j \delta_j \geq \max_j \pi_j$. Now let j be the packet incurring the maximum flow time in the schedule obtained by FIFO. Since $r_j \geq r_{b(j)}$ (by definition of FIFO), we have

$$R_{b(j)} - r_j = r_{b(j)} + \delta_{b(j)} - \pi_{b(j)} - r_j \leq \delta_{b(j)} \tag{9}$$

Using Lemma 4.1 and (9), we get

$$\begin{aligned}F_j = C_j - r_j &\leq R_{b(j)} - r_j + \frac{\gamma}{\gamma^*}\sum_{i\in P(j)} \pi_i \\ &\leq \delta_{b(j)} + \frac{\gamma}{\gamma^*}\sum_{i\in P(j)} \pi_i \\ &\leq \max_i F_i^* + \frac{\gamma}{\gamma^*}\cdot |P(j)| \cdot \max_i F_i^* \\ &\leq \left(1 + \frac{\gamma}{\gamma^*} m\right)\max_i F_i^*.\end{aligned}$$

□

5. APPROXIMATION ALGORITHMS FOR TOTAL FLOW TIME

In this section we study FSUM-WGP. We first observe that σ-FIFO is a not a σ-speed optimal algorithm for FSUM-WGP, for any constant σ.

PROPOSITION 5.1. *There is no $\sigma \geq 1$ such that σ-FIFO is a σ-speed optimal algorithm for* FSUM-WGP.

PROOF. Consider an instance where the network is a path on $m+1$ nodes, with the sink at one end, and $d_I = m+1$. There are m packets. Packet 1 has a release date of 0 and is released at distance m from the sink. Packet j, for $j = 2, \ldots, m$, has a release date of $j-1$ and is released at distance 1 from the sink. If packets are processed one at a time, in the order $2, 3, \ldots, m, 1$, the total flow time is $O(m)$. On the other hand, because σ-FIFO gives priority to packet 1, which is completed at time m/σ, none of the other packets will be completed before time m/σ. In particular, packets $2, 3, \ldots, \lfloor m/2\sigma \rfloor$ will each have a flow time of at least $m/2\sigma$, implying a total flow time of $\Omega(m^2/\sigma^2)$ for σ-FIFO. The approximation ratio is thus at least $\Omega(m/\sigma^2)$, which can be made arbitrarily large for any fixed σ (in fact, for any $\sigma = o(m^{1/2})$). □

We introduce an algorithm that we call INTERLEAVED SRPT and prove that a constant-factor speed augmentation is enough to enable this algorithm to outperform the optimal total flow time of the original instance. The algorithm is based on a well-known scheduling algorithm, the *Shortest Remaining Processing Time* rule (SRPT) [Schrage 1968]. We first describe SRPT in the context of WGP (Algorithm 3).

Algorithm 3 SRPT

```
for $k = 0, 1, 2, \ldots$ do
    At time $t = k/\sigma$, let $1, \ldots, m'$ be the available packets in order of non-decreasing distance to the sink (that is, $d(x_1^t, s) \leq d(x_2^t, s) \leq \ldots \leq d(x_{m'}^t, s)$)
    for $j = 1$ to $m'$ do
        Send $j$ to the next hop along an arbitrary shortest path from $x_j^t$ to the sink, unless this creates interference with a packet $j'$ with $j' < j$
    end for
end for
```

Every iteration k in the algorithm corresponds to a round of the schedule. We notice that this algorithm is a dynamic-priority algorithm, in the sense that the ordering in which packets are scheduled can change from round to round. As such, it cannot be cast in the priority-based framework presented in the previous section.

We also notice that, if $\delta_j < \gamma^*$ for each packet $j \in J$ (that is, when all packets are released inside the critical region), then WGP reduces to a single machine scheduling problem with preemption; compare with the proof of Lemma 4.2. The problem of minimizing total flow time is in that case equivalent to the single machine scheduling problem with the same objective, allowing preemption and with jobs having release times: $1|r_j, \text{pmtn}|\sum_j F_j$ in terms of classical scheduling notation [Graham et al. 1979]. Schrage [1968] showed that SRPT solves the latter problem to optimality.

In fact he proved it for the problem of minimizing total completion times, which, if solved to optimality, is equivalent to that of minimizing total flow time.

Consider a schedule generated by σ-speed SRPT, that is, every round is executed in time $1/\sigma$. It will be convenient to refer to round k, corresponding to the time interval $[k/\sigma, (k+1)/\sigma)$, as "round k/σ".

We denote the ith packet to arrive at the sink in this schedule as $p(i)$, for $1 \leq i \leq m$. We define a *component* as a maximal set T of packets with the following properties:

(1) There is an index a such that $T = \{p(a), p(a+1), \dots, p(a+|T|-1)\}$;

(2) For all $i = 1, 2, \dots, |T|-1$, $C_{p(a+i)} \leq C_{p(a+i-1)} + \gamma/\sigma$;

That is, a component is a maximal set of packets arriving subsequently at the sink, each within time γ/σ of the previous packet. It follows from the definition that the set J of all packets can be partitioned into components $T_1, \dots, T_\ell$, for some ℓ.

LEMMA 5.2. *For any component T,* $\min_{j \in T} C_j = \min_{j \in T}(r_j + \delta_j/\sigma)$.

PROOF. Consider the partition of the packet set J into components $T_1, \dots, T_\ell$. The components are ordered so that $\max_{j \in T_i} C_j < \min_{k \in T_{i+1}} C_k$ for each i; by definition of a component such an ordering exists.

Let $S(i) = \cup_{h=i}^{\ell} T_h$, for $1 \leq i \leq \ell$. We define $\overline{t}_i := \min_{j \in S(i)}(r_j + \delta_j/\sigma)$, the earliest possible completion time of any packet in $S(i)$, and $\underline{t}_i := \max\{r_j : j \in S(i) \text{ and } r_j + \delta_j/\sigma = \overline{t}_i\}$, the maximum release date of a packet in $S(i)$ with earliest possible completion time $\overline{t}_i$. Consider the following set of packets, for any t with $\underline{t}_i \leq t \leq \overline{t}_i$:

$$M_i(t) = \{j \in S(i) : r_j \leq t < C_j \text{ and } d(x_j^t, s) \leq d(x_k^t, s) \text{ for all } k \in S(i)\},$$

That is, $M_i(t)$ is the set of packets in $S(i)$ that have been released but not completed, and that have minimum distance from the sink among the packets in $S(i)$. Note that $M_i(t)$ is nonempty for $t \in [\underline{t}_i, \overline{t}_i]$, because no packet in $S(i)$ arrives at the sink before round $\overline{t}_i$.

The proof of the lemma relies on the following claim.

Claim. *For $i = 1, 2, \dots, \ell$ and for any round $t \in [\underline{t}_i, \overline{t}_i]$ there exists $j \in M_i(t)$ such that $d(x_j^t, s) \leq \sigma(\overline{t}_i - t)$.*

Suppose the claim holds. Then taking $t = \overline{t}_i$ implies that for each $i = 1, \dots, \ell$, there is a packet $j \in M_i(\overline{t}_i)$ which arrives at the sink in round $\overline{t}_i = \min_{k \in S(i)} r_k + \delta_k/\sigma$. As a consequence $j \in T_i$, and $\overline{t}_i = r_j + \delta_j/\sigma$, which proves the lemma.

Proof of Claim. The claim trivially holds for $t = \underline{t}_i$, because some packet $j \in S(i)$ with earliest possible arrival time $\overline{t}_i$ is released in round $\underline{t}_i$, hence $\underline{t}_i + d(x_j^{\underline{t}_i}, s)/\sigma = r_j + \delta_j/\sigma = \overline{t}_i$.

Suppose that during each round $t \in [\underline{t}_i, \overline{t}_i]$ some packet in $M_i(t)$ is sent towards the sink. Then the inequality of the claim follows, as it is true when $t = \underline{t}_i$, and both sides decrease by one unit after t is increased by $1/\sigma$ (that is, after every round).

Otherwise, there must be a latest round $t' \in [\underline{t}_i, \overline{t}_i]$ in which no packet in $M_i(t')$ is sent towards the sink. Then by definition of SRPT there must be a packet $k \in J \backslash S(i)$ which is sent, and a packet $j \in M_i(t')$ for which $d(x_j^{t'}, x_k^{t'}) \leq d_I + 1$. Since j is not sent during round t', we also have $d(x_j^{t'+1/\sigma}, x_k^{t'}) \leq d_I + 1$. Additionally, $d(x_k^{t'}, s)/\sigma \leq C_k - t'$ because otherwise k could not reach the sink by time C_k. Now for each round $t \in [t' + 1/\sigma, \overline{t}_i]$ a packet in $M_i(t)$ is sent. In particular, there must be a packet from the set $\cup_{t \in [t'+1/\sigma, \overline{t}_i]} M_i(t)$, call it j', that arrives at the sink no later than j would arrive if j were always sent from round $t' + 1/\sigma$ onwards. We have

$$C_{j'} \leq (t' + 1/\sigma) + \left(d(x_j^{t'+1/\sigma}, x_k^{t'}) + d(x_k^{t'}, s)\right)/\sigma \leq C_k + (d_I + 2)/\sigma = C_k + \gamma/\sigma.$$

That is, packet $j' \in S(i)$ arrives at most γ/σ time units after packet $k \in J \setminus S(i)$, which contradicts the fact that j' and k are in different components. Thus this case never occurs and the claim holds. $\square$

We now describe Interleaved SRPT. The algorithm partitions the set of packets J in two subsets, $J^{\text{in}} := \{j \in J : \delta_j < \gamma^*\}$ and $J^{\text{out}} := \{j \in J : \delta_j \geq \gamma^*\}$. The two subsets are scheduled in an interleaved fashion using SRPT. The pseudocode is given as Algorithm 4.

Algorithm 4 Interleaved SRPT

```
**Inizialization**: $c := 1$
**loop**
  **if** $c \neq 0 \pmod 5$ **then**
    execute one round of SRPT on the set $J^{\text{out}}$
  **else**
    execute one round of SRPT on the set $J^{\text{in}}$
  **end if**
  $c := c + 1$
**end loop**
```

In the performance analysis of Interleaved SRPT we use the following lower bound on the sum of optimal completion times of subsets of jobs in J^{out}, which is proved similarly to Lemma 4.2.

Lemma 5.3. *Let $S \subseteq J^{\text{out}}$. If C_j^* denotes the completion time of packet j in any feasible schedule, we have*

$$\sum_{j \in S} C_j^* \geq \sum_{i=0}^{|S|-1} (C_S + i\gamma^*)$$

where $C_S := \min_{j \in S} r_j + \delta_j$.

Proof. Because all packets in S are at distance at least γ^* from the sink, each of them needs to be transmitted in the critical region for γ^* (not necessarily consecutive) rounds, thus blocking the transmission of any other packet inside the region. Also, the earliest packet in S to enter the critical region cannot enter it before time $C_S - \gamma^*$, so the ith packet to arrive at the sink cannot reach it before time $C_S - \gamma^* + (i+1)\gamma^*$. Thus, the total completion time of packets in S is at least

$$|S| \cdot (C_S - \gamma^*) + \sum_{i=0}^{|S|-1} (i+1)\gamma^* = \sum_{i=0}^{|S|-1} (C_S + i\gamma^*) \,.$$

□

We denote the σ-speed version of Interleaved SRPT as σ-ISRPT.

Theorem 5.4. *For $\sigma \geq \gamma/\gamma^* + 1$, σ-ISRPT is optimal for* Fsum-Wgp.

Proof. Let C_j be the completion time of packet j in a σ-speed ISRPT schedule, and let C_j^* be the completion time of packet j in any feasible (possibly optimal) unit speed schedule. We prove the theorem by showing that $\sum_{j \in J^{\text{in}}} C_j \leq \sum_{j \in J^{\text{in}}} C_j^*$ and $\sum_{j \in J^{\text{out}}} C_j \leq \sum_{j \in J^{\text{out}}} C_j^*$.

Consider first the packets in J^{in}. Since we are executing ISRPT at speed σ, and the set J^{in} is considered once every σ iterations, we have that during every time interval $[t, t+1)$, $t = 0, 1, \ldots$, a round of SRPT is executed on the set J^{in}. So the completion times of the packets in J^{in} are not worse than those that would be obtained by running SRPT with unit speed on J^{in} alone. On the other hand, inside the critical region the gathering problem is nothing else than the scheduling problem $1|r_j, \text{pmtn}|\sum_j C_j$, meaning that SRPT is optimal. It follows that $\sum_{j \in J^{\text{in}}} C_j \leq \sum_{j \in J^{\text{in}}} C_j^*$.

Consider now the packets in J^{out}. Because in the *first* four out of every five rounds of σ-ISRPT this set is scheduled using SRPT, the completion time of each packet in J^{out} is not larger than the completion time of the same packet in a $(\sigma - 1)$-speed SRPT (non-interleaved) schedule of J^{out}:

$$C_j \leq \overline{C}_j \text{ for all } j \in J^{\text{out}}, \tag{10}$$

where $\overline{C}_j$ is the completion time of j in a $(\sigma - 1)$-speed SRPT schedule of J^{out}. Consider any component T in this latter schedule. By Lemma 5.2 and the definition of component

$$\sum_{j \in T} \overline{C}_j \leq \sum_{0 \leq i < |T|} (C_T + \frac{1}{\sigma - 1} i\gamma). \tag{11}$$

On the other hand by Lemma 5.3,

$$\sum_{j \in T} C_j^* \geq \sum_{0 \leq i < |T|} (C_T + i\gamma^*). \tag{12}$$

For any $\sigma \geq \gamma/\gamma^* + 1$, combining (10), (11) and (12) implies that $\sum_{j \in T} C_j \leq \sum_{j \in T} C_j^*$. The result follows after summing over all the components, and subtracting $\sum_{j \in J} r_j$ from both sides. □

As in the previous section, we use that $2 \leq \gamma/\gamma^* \leq 4$ for all d_I, and that $\gamma/\gamma^* = 3$ when $d_I = 1$, to obtain the main result of this section.

Corollary 5.5. 5-ISRPT *is a* 5*-speed optimal algorithm for* Fsum-Wgp. *When* $d_I = 1$, 3-SRPT *is a* 3*-speed optimal algorithm for* Fsum-Wgp.

Proof. When $d_I = 1$, one has $\gamma^* = 1$ and the set J^{in} is empty, so there is no need for interleaving and it suffices to run SRPT with speed $\gamma/\gamma^* \leq 3$. □

As a further corollary of the preceding analysis, we obtain a bound on the approximation ratio of Wgp when the objective is the minimization of the sum of completion times.

Corollary 5.6. *There is a 5-approximation algorithm for* Csum-Wgp*. When $d_I = 1$, there is a 3-approximation algorithm for* Csum-Wgp*.*

Proof. Notice that the analysis of the above theorem also yields that

$$\sum_{j \in J} C_j \leq \sum_{j \in J} C_j^*,$$

where C_j is the completion time of packet j in the schedule generated by 5-ISRPT. We can now simulate the schedule generated by 5-ISRPT by running it at a lower speed: whatever 5-ISRPT does at time t, a unit-speed algorithm can do at time $5t$. The schedule can be constructed online and clearly it respects the release dates. If C'_j is the completion time of packet j in the new schedule, we obtain

$$\sum_{j \in M} C'_j = \sum_{j \in M} 5C_j \leq 5 \sum_{j \in M} C_j^*.$$

A similar analysis shows a 3-approximation when $d_I = 1$. □

6. CONCLUSIONS

We considered the wireless gathering problem with the objectives of minimizing the maximum flow time (Fmax-Wgp) and total flow time of data packets (Fsum-Wgp). We showed that the simple on-line algorithm FIFO and a slightly adapted version of SRPT have favorable behavior, although the problems are both extremely hard to approximate in general, even on geometric graphs. For Fmax-Wgp, augmenting the transmission rate by a factor of 5 allows FIFO to achieve a cost bounded by that of an optimal solution. A similar result holds for Fsum-Wgp, if we resort to the ISRPT algorithm.

For both problems, it is an open question whether optimality can be achieved by augmenting the transmission rate by a factor smaller than 5. With respect to Fsum-Wgp, we have not been able to prove a resource augmentation result for SRPT, but only for the modified algorithm with interleaving. It would be interesting to show a similar result for SRPT.

Another interesting set of questions concerns resource augmentation by allowing the algorithms to use extra frequencies, meaning that more than several data packets can be sent simultaneously using different channels. It would be interesting to compare the relative power of increasing the channels with that of increasing the speed.

Another open problem of a more theoretical nature is to settle the complexity of Fmax-Wgp and Fsum-Wgp when restricted to unit disk graphs. We conjecture that both problems remain NP-hard in that case, but even then, it would be interesting to determine whether the unit disk graph assumption allows for better approximation guarantees.

For the minimization of the completion time (Cmax-Wgp), the existence of a polynomial time approximation scheme is still open. It is known that no algorithm that uses shortest paths to route the data packets to the sink can give an improvement over the currently best approximation ratio. It appears challenging to design and analyze congestion avoiding algorithms with better ratios.

Finally, the gathering problem can be generalized to an end-to-end problem, where packets have different origins and destinations. We observe that the lower bounds used in our proofs on the cost of an optimal solution do not extend to this setting, because in that case multiple packets can reach their destinations in a single round. Extending our positive results to this setting is thus an interesting open question.

ACKNOWLEDGMENTS

We thank the anonymous referees for several suggestions that helped us improve the presentation of the paper, as well as its content.

REFERENCES

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. 2002. Wireless sensor networks: a survey. *Computer Networks 38,* 4, 393–422.

Bar-Yehuda, R., Goldreich, O., and Itai, A. 1992. On the time-complexity of broadcast in multi-hop radio networks: an exponential gap between determinism and randomization. *Journal of Computer and Systems Sciences 45,* 1, 104–126.

Bar-Yehuda, R., Israeli, A., and Itai, A. 1993. Multiple communication in multihop radio networks. *SIAM Journal on Computing 22,* 4, 875–887.

Bender, M. A., Chakrabarti, S., and Muthukrishnan, S. 1998. Flow and stretch metrics for scheduling continuous job streams. In *Proc. 9th Symp. on Discrete Algorithms.* SIAM, Philadelphia, PA, 270–279.

Bermond, J.-C., Galtier, J., Klasing, R., Morales, N., and Pérennes, S. 2006. Hardness and approximation of gathering in static radio networks. *Parallel Processing Letters 16,* 2, 165–183.

Bonifaci, V., Korteweg, P., Marchetti-Spaccamela, A., and Stougie, L. 2008. An approximation algorithm for the wireless gathering problem. *Operations Research Letters 36,* 5, 605–608.

Borodin, A. and El-Yaniv, R. 1998. *Online Computation and Competitive Analysis.* Cambridge University Press, Cambridge.

Chan, H.-L., Lam, T. W., and Liu, K.-S. 2006. Extra unit-speed machines are almost as powerful as speedy machines for competitive flow time scheduling. In *Proc. 17th Symp. on Discrete Algorithms.* SIAM, Philadelphia, PA, 334–343.

Chekuri, C., Goel, A., Khanna, S., and Kumar, A. 2004. Multi-processor scheduling to minimize flow time with epsilon resource augmentation. In *Proc. 36th Symp. on Theory of Computing.* ACM, New York, NY, 363–372.

Clark, B. N., Colbourn, C. J., and Johnson, D. S. 1990. Unit disk graphs. *Discrete Mathematics 86,* 1-3, 165–177.

Florens, C., Franceschetti, M., and McEliece, R. J. 2004. Lower bounds on data collection time in sensory networks. *IEEE Journal on Selected Areas in Communications 22*, 1110– 1120.

Gargano, L. and Rescigno, A. A. 2006. Optimally fast data gathering in sensor networks. In *Proc. 31st Symp. on Mathematical Foundations of Computer Science.* Springer, Berlin, 399–411.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math. 5*, 287–326.

Kalyanasundaram, B. and Pruhs, K. 2000. Speed is as powerful as clairvoyance. *Journal of the ACM 47,* 4, 617–643.

Ko, C. W. and Shepherd, F. B. 2003. Bipartite domination and simultaneous matroid covers. *SIAM Journal on Discrete Mathematics 16,* 4, 517–523.

Korteweg, P. 2008. Online gathering algorithms for wireless networks. Ph.D. thesis, Technische Universiteit Eindhoven.

Anil Kumar, V. S., Marathe, M. V., Parthasarathy, S., and Srinivasan, A. 2004. End-to-end packet-scheduling in wireless ad-hoc networks. In *Proc. 15th Symp. on Discrete Algorithms*, J. I. Munro, Ed. SIAM, Philadelphia, PA, 1021–1030.

Anil Kumar, V. S., Marathe, M. V., Parthasarathy, S., and Srinivasan, A. 2005. Algorithmic aspects of capacity in wireless networks. In *Proc. Conf. on Measurement and Modeling of Computer Systems*. ACM, New York, NY, 133–144.

Leonardi, S. and Raz, D. 2007. Approximating total flow time on parallel machines. *Journal of Computer and Systems Sciences 73,* 6, 875–891.

Leung, J. Y.-T., Ed. 2004. *Handbook of Scheduling*. CRC Press, Boca Raton, FL.

Linial, N. 1992. Locality in distributed graph algorithms. *SIAM Journal on Computing 21,* 1, 193–201.

Pahlavan, K. and Levesque, A. H. 1995. *Wireless information networks*. Wiley, New York, NY.

Pelc, A. 2002. Broadcasting in radio networks. In *Handbook of Wireless Networks and Mobile Computing*. Wiley, New York, NY, 509–528.

Ramanathan, S. and Lloyd, E. L. 1993. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transactions on Networking 1,* 2, 166–177.

Schmid, S. and Wattenhofer, R. 2006. Algorithmic models for sensor networks. In *Proc. 20th Int. Parallel and Distributed Processing Symp.* IEEE, Washington, DC.

Schrage, L. 1968. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research 16,* 3, 687–690.

Torng, E. and McCullough, J. 2008. SRPT optimally utilizes faster machines to minimize flow time. *ACM Transactions on Algorithms 5,* 1.

Received Month Year; revised Month Year; accepted Month Year