SCHEDULING AND RESOURCE ALLOCATION IN WIRELESS SENSOR NETWORKS

by

YOSEF ALAYEV

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment
of the requirements for the degree of Doctor of Philosophy,
THE CITY UNIVERSITY OF NEW YORK
2014

This manuscript has been read and accepted for the
Graduate Faculty in Computer Science in satisfying of the
dissertation requirement for the degree of Doctor of Philosophy.

	Amotz Bar-Noy
Date	Chair of Examining Committee
	Robert Haralick
Date	Executive Officer

Theodore Brown

Noson Yanofsky

Stathis Zachos

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

SCHEDULING AND RESOURCE ALLOCATION IN WIRELESS SENSOR NETWORKS

by

Yosef Alayev

Adviser: Doctor Amotz Bar-Noy

In computer science and telecommunications, wireless sensor networks are an active research area. Each sensor in a wireless sensor network has some pre-defined or on demand tasks such as collecting or disseminating data. Network resources, such as broadcast channels, number of sensors, power, battery life, etc., are limited. Hence, a schedule is required to optimally allocate network resources so as to maximize some profit or minimize some cost. This thesis focuses on scheduling problems in the wireless sensor networks environment. In particular, we study three scheduling problems in the wireless sensor networks: broadcast scheduling, sensor scheduling for area monitoring, and content distribution scheduling. For each problem the goal is to find efficient scheduling algorithms that have good approximation guarantees and perform well in practice.

Acknowledgements

There are a number of people that I want to thank for their help, support, and guidance during the course of my graduate work as well as the writing process of this thesis. First of all I want to thank my graduate mentor and advisor Dr. Amotz Bar-Noy for his patience and guidance during this process. He taught me not only about research in computer science, but also about being a responsible and generous person.

I acknowledge and thank all the present members of my Thesis Advisory and Defense Committee: Theodore Brown, Noson Yanofsky, and Stathis Zachos for their support and encouragement during the course of my work.

I want to thank all of my coauthors including Thomas La Porta, Matthew P. Johnson, Lance Kaplan, Fangfei Chen, Yun Hou, and Kin Leung with whom I had a privilege of collaborating. Thanks for providing valuable feedback as well as technical and intellectual support during the course of my studies.

Additionally, I want to thank all of my professors at The Graduate Center and at Queens College, especially Dr. Bojana Obrenic for her encouragement and intellectual support.

I thank my family. Thank you Mom and Dad for everything that I am and for giving me the tools to build my life. You are always proud of me and believe that I can do anything that I put my mind to. Thanks to my brother Vadim with his wife Natalie and my brother Gavriel for their moral support.

Finally, I thank my wife Anna for her moral and emotional support and for pushing me to finish my graduate work.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Scheduling Overview

Scheduling is an allocation of scarce resources to activities over time. Problems of scheduling or optimal resource allocation are motivated by questions that arise in project scheduling, production planning, computer control, broadcasting, routing data, maintenance scheduling, and so on. Although the field is relatively new and dates back to early fifties, an enormous amount of literature has been created with lots of open problems that are still under investigation by researchers. Models in the scheduling area have become highly standardized, in order to be able to keep track of a vast amount of scheduling problems that have optimal solutions or problems that are still open. The models address scheduling of n jobs (the activities such as pages or clients) on m machines (the resources such as broadcast channels or servers). A machine can process no more than one job at a time. The goal is then to allocate machines to jobs, optimizing some objective function in doing so. The objective function may have something to do with maximizing some profit or minimizing some cost of a constructed schedule. For a survey of the field see [32, 39, 10]. For an annotated bibliography of some influential scheduling papers see [35].

Scheduling can be periodic—where pattern repeats every T timeslots for some fixed T [34], or non-periodic—where jobs are scheduled once, which is more standard. In both types of scheduling, the objective may be either optimal throughput—where the goal is to schedule as many profitable jobs as possible, or load—where the goal is to distribute workloads across multiple machines. Scheduling may be offline—where the jobs are already known to the scheduler at the beginning of the algorithm, or online—where the scheduler needs to schedule jobs as they arrive. Scheduling may be preemptive—where jobs are allowed to stop processing in the middle and resumed their process in a later time, or non-preemptive—where jobs once scheduled must run until completion. Migration may be allowed, in which case processing a job can be stopped and resumed later on a different machine. Schedulers may be centralized—where the computer application controlling the scheduling process is on a single central computer, or distributed—where the scheduler is distributed across many computers. Machines used in scheduling may be identical—where machines operate at the same speed per job, or different—where machines may have their own speeds or machine speeds may be job dependant.

1.1.1 Formal Definitions

Suppose that m machines $M_j(j = 1, \ldots, m)$ have to process n jobs $J_i(i = 1, \ldots, n)$. A schedule allocates to each job one or more time intervals at one or more machines. We assume that each machine can process at most one job at a time and that each job can be processed on at most one machine at a time. Lawler [32] introduced a 3-field notation $\alpha|\beta|\gamma$ for problem classification that can capture various job, machine, and scheduling characteristics. The fields are used to specify *machine environment* (α), *job characteristics*

(β), and an *optimality criterion* (γ).

Machine environment: The simplest environment is the *single machine* on which jobs J_i, each consisting of a single operation, have to spend processing times p_i. In a *parallel-machine* environment job J_i has to spend a given time on any of the m machines. The machines can be *identical* represented by P, in which case the machines operate at the same speed; *uniform* (or *related*) represented by Q, in which case each machine has its own speed; and *unrelated* represented by R, in which case the speed of the machine is job-dependant. *Open shops*, *flow shops*, and *job shops* are m-machine environments. In all three environments each job consists of several operations. Each operation of a job has to be executed on a designated machine; no job can undergo more than one operation at a time. In *job shops* the order in which the operations of a job have to be executed is fixed. In *flow shops* the order is fixed and is the same for all jobs. In *open shops* the order is free and is up to the scheduler.

Job characteristics: This field specifies the possibility of allowing *preemption* (or job splitting) and specifying the precedence constraints. Once the preemption is allowed, then the operation can be interrupted and resumed later on; otherwise once the operation is started it must by carried out until its completion without interruption. A precedence constraint specifies that a certain job cannot start before another is finished. The precedence relations between jobs can be given as an acyclic directed graph $G = (V, E)$. Job availability can be restricted by imposing release time r_i, before which the job cannot be started, and deadline time d_i, before which the jobs must be completed. Jobs may specify processing times p_i and weights w_i. In some scheduling applications, sets of jobs must be grouped into batches, which can be specified in job characteristics field.

Optimality criteria: A feasible schedule is an allocation of jobs to time intervals on the machines such that all restrictions and constraints are met. The optimality criterion can be a function of the job completion times $C_1, \ldots, C_n$. Criteria that are common are *maximum completion time (makespan)* $C_{max} = \max_i C_i$ or *total completion time* $\sum_i C_i$. We can compute lateness, tardiness, and earliness for jobs that have deadline times. They are $L_i = C_i - d_i$, $T_i = \max(0, C_i - d_i)$, and $E_i = \max(0, d_i - C_i)$ respectively. Important criteria involving these can be *maximum lateness* $L_{max} = \max_i L_i$, *total tardiness* $\sum_i T_i$, and *total earliness* $\sum_i E_i$. If we let the unit penalty, U_i, to be either 0 (not late: $C_i \leq d_i$) or 1 (late), then we can have a criteria of *total late jobs* $\sum_i U_i$, or maximum throughput $\sum_i (1 - U_i)$. For all these we can have weighted version, if each job has a weight of w_i. For example, we can have a weighted maximum throughput $\sum_i w_i (1 - U_i)$.

1.1.2 Scheduling applications

Consider, for example, a broadcast system such as a system that broadcasts information about the stocks on a single broadcast channel. This single broadcast channel (one machine) must be dedicated to scheduling the broadcasts of many pages, each containing updated information about a single stock. The bandwidth of the channel has to be divided among those pages. Suppose then that some stocks are more popular than others and may require more frequent updates. The problem is to allocate fractions of the channel to individual pages in such a way that the average response time for getting information about any stock is minimized.

Now consider a wireless sensor network employed in rescue applications. Data must be delivered to mobile clients en-route, as they travel toward destination. The data can be delivered to the mobile clients as they pass within range of wireless base stations. As a restriction, assume that one client can communicate with any one given base station at a time. Consider for instance a scenario where building blueprints are delivered to firefighters while they are responding to multiple alarms. The problem can be cast as a parallel-machine scheduling problem.

Many applications and systems exist in scheduling. Examples of applications for periodic scheduling are: teletext systems [11, 12], broadcast disks [4], windows-scheduling for media on-demand [19], maintenance scheduling [13, 17], sensor networks [36, 56, 62, 22, 28]. Examples of applications for non-periodic scheduling are: real-time systems [18, 21], timeslot assignment to mobile clients in WSN [25], space mission scheduling [33], military systems [40], to name just a few. The general resource allocation problems of knapsack [51] or multiple knapsack [15], and bin-packing [30] can be thought as special scheduling problems with unrestricted time constraints.

1.2 Wireless Sensor Networks

Research in wireless sensor networks is an active area. Each sensor in a wireless sensor network has some pre-defined or on demand task such as collecting or dissimilating data. Network resources at both the individual node level as well as the network level are constrained. Several performance challenges in wireless sensor networks are reducing data or event obsoleteness, increasing lifetime of the network or data throughput, etc. These performance criteria can be formulated as maximizing some profit or minimizing some costs. Thus, sensor networks introduce new resource allocation problems where the goal is to schedule network resources optimally so as to maximize some profit or minimize some cost. The problems we study in this dissertation are scheduling problems and their algorithms as applied to sensor network environments. Such problems have been previously studied in simplified models. We study more complex problems focussing on applications such as monitoring, surveillance, and applications such as rescue missions. The first two problems that we study are examples of periodic scheduling where tasks need to be performed infinitely often. The last problem that we study is an example of non-periodic scheduling where clients need to pull data from stationary access points during a specific time.

1.3 Topics

1.3.1 Broadcast Scheduling of Info-Pages to Sensors

In sensor networks applied to monitoring applications, individual sensors may perform preassigned or on-demand tasks, or *missions*. Data updates (info-pages) may be sent to sensors from a command center, via a time-division broadcast channel. Sensors are normally put in sleep mode when not actively listening, in order to conserve energy of their batteries. Hence, a schedule is required that specifies when sensors should listen for updates and when they should sleep. The performance of such a schedule is evaluated based on data-related costs and sensor-related costs. *Data-related costs* reflect the obsoleteness of current sensor data, or the delay while sensors wait for updated instructions. *Sensor-related costs* reflect the energy that sensors consume while accessing the broadcast channel and while switching between the active and sleeping modes (rebooting). Our goal is to find a schedule with the minimum total cost. Previous related work has explored data-related costs, but listening cost has been addressed only under the assumption that the rebooting operation is free. We study a problem with a new cost model, which recognizes the cost of sensor rebooting.

1.3.2 Single Sensor Scheduling Problem with Refocusing Delays

For the sensor scheduling we study a problem in which a single sensor is scheduled to observe sites periodically, motivated by applications in which the goal is to maintain up-to-date readings for all the observed sites. In the existing literature, it is typically assumed that the time for a sensor switching from one site to another is negligible. This may not be the case in applications such as camera surveillance of a border, however, in which the camera takes time to pan and tilt, refocusing itself to a new geographical location. We study a sensor scheduling problem with refocusing delay constraints. We prove the problem to be NP-hard and then study a special case in which refocusing is proportional to some Euclidian metric. We give a lower bound on the optimal cost for the scheduling problem. Finally, we provide and experimentally evaluate several heuristic algorithms, some of them based on this computed lower bound.

1.3.3 Content Distribution Scheduling

For the content distribution scheduling we study a data dissemination scenario in which data items are to be transmitted to mobile clients via one of the stationary data access points (APs) that the clients pass by en route to their destinations. The scheduler dedicates sequences of consecutive timeslots of an AP to downloading a data item to a client during the time window in which it is in range, which corresponds to assigning a job (the client's download) to a machine (the AP) among many. The transmission rate chosen for each assignment partly corresponds to setting a machine's speed, but it also has subtler effects. The APs may control transmission power to tune its transmission range making sure that no interference occurs with neighboring APs' transmissions. The problem is a generalization of an already NP-hard parallel-machine scheduling problem in which jobs' release times and deadlines depend on the machine to which they are assigned. We study this joint timeslot, power control, and rate assignment problem formally and apply both new algorithms and adaptations of existing algorithms to it. We evaluate these algorithms through simulations to show that our proposed algorithms achieve near-optimal throughput.

1.4 Organization

The rest of the dissertation is divided into three parts. Chapter 2 presents the work on broadcast scheduling problem. Sensor scheduling for the geographical area monitoring is subject of Chapter 3. Both of these chapters are on periodic scheduling. The subject of Chapter 4 is content distribution scheduling that is non-periodic. We conclude with Chapter 5.

Chapter 2

Broadcasting Info-Pages to Sensors: Efficiency vs. Energy Conservation

This chapter[1] presents a problem of broadcast scheduling in sensor networks employed in monitoring applications. The goal is to construct a schedule of updates to individual sensors while at the same time optimizing schedule efficiency and energy conservation.

2.1 Introduction and Motivation

In monitoring applications, nodes in a sensor network typically perform multiple dynamic missions, recording audio, video, magnetic, seismic, chemical, or infrared light data. The mission of a camera at a particular location could be general (e.g. to observe any movement that occurs) or prompted by specific information (e.g. to locate a person with a scar on his face). Missions could be pre-assigned, initiated by sensors, or initiated by a command center that controls all the sensors. We consider a setting in which a command center

[1]The work of this chapter is published in [7] and [8].

communicates with sensors via a shared broadcast channel, in order to send them *descriptions* or mission instructions. Since missions may change, sensors must listen continually for updated mission descriptions or *info-pages.*

Since sensors typically run on finite batteries, and since sensors must operate for long intervals of time without having these batteries replaced, energy conservation is an important goal. Sensors are typically placed in sleep mode when not actively listening. However, if a sensor happens to be sleeping while new instructions are broadcast, its current instructions become obsolete, and remain so for some time. Our goal is to schedule the delivery of pages in such a way so as to minimize the information obsoleteness, measured by time elapsed between the broadcast of a page and execution of the updated mission, and at the same time maximize the life of the battery on which the sensor operates.

For instance, consider a case of two sensors that are required to execute two missions. The first sensor executes a mission whose information is contained in page A and the second sensor executes a mission whose information is contained in page B. Pages A and B are broadcast via a shared channel. Assuming that both missions are equally likely to be executed, the schedule should not prefer one page to the other, but should broadcast the two pages in a round-robin fashion. This leads to a periodic schedule $D_1 : (AB)^*$, with a period of 2.

It is easy to construct a case in which page A has priority over page B. In this case, page A may be transmitted more often. A disadvantage of this method is found in the obsoleteness of data owned by the second sensor, which occurs because page B is broadcast less often.

Our example takes into consideration only data-related costs. When sensor-related costs are admitted in the model, a different schedule may become optimal. For instance, if the first sensor, while *listening,* (operating in the active state) incurs a cost high enough, then it may be advantageous to conserve its battery and broadcast A less often even if it is of higher priority.

Consider now the cost of *rebooting*, which accounts for energy losses caused by switching between the active and sleep modes. If this cost is introduced into the model, its effects on the performance tend to oppose those of the listening cost. For example, if rebooting the first sensor is very costly, then it may be appropriate to reform the schedule to have it receive more pages even if is listening cost is high. This would reverse the effect described before caused by the high listening cost to the sensor with the larger popularity value.

Construction of an optimal schedule depends nontrivially on the specific quantitative relationships among all the considered costs. Where obsoleteness costs are not exceedingly high, a sensor should save its energy by going into sleep mode. However, if the rebooting operation is too expensive and the sensor must be awake again soon, staying in the active mode may be more efficient—even if there are no updated instructions to listen for. The optimal schedule depends on the precise values of all the costs involved. Our goal is to find the right balance between the data-related and sensor-related costs, scheduling updates in such a way as to minimize the obsoleteness of data and at the same time preserve energy to the extent possible.

In this chapter we first review related work of broadcast scheduling (Section 2.2.) We then formulate a new model that incorporates data-related and sensor-related costs (Section 2.3.) In this model, we provide an optimal solution in the setting with one sensor (Section 2.4.) Based on this solution, we formulate and give a solution to a non-linear mathematical program, whose optimal solution serves as a lower bound for the many-sensor setting (Section 2.5.) Finally, we design heuristics based on greedy rules and compare their performance with a simple round-robin schedule (Section 2.6.)

2.2 Related Work

Broadcast systems are push-based delivery systems, in which clients do not request data, but listen on a shared broadcast channel. The server pushes data (info pages) to the clients (sensors), according to a schedule that makes no provisions for any incoming requests. Early research in data broadcast problems [11][12][17] focuses on specialized settings, where all pages have unit length and are delivered on a single broadcast channel over discrete time.

Ammar and Wong [11][12] consider the problem of minimizing the average response time in Teletext Systems, which is equivalent to the Broadcast Disks [5] problem; they give a lower bound and prove the existence of a periodic schedule. Bar-Noy *et al.*[17] prove that the problem with broadcast costs is NP-hard and give a constant-factor approximation algorithms for the basic model. They also define a general model that unifies several important previous models, formulate a non-linear program to provide a lower bound, and present an efficient algorithm for finding a near-optimal solution.

Busy-waiting consumes energy. A common way to minimize this consumption is to design schedules that allow users to spend most of their time turned off. For example, Bluetooth's Sniff Mode, Hold Mode, and Park Mode allow a client to sleep except during some predefined periodic intervals [1]. Bar-Noy *et al.*[17] incorporate a client listening cost into their model, but they allow a client to go into sleeping mode and reboot at no cost. Khanna and Zhou [38] show how to use indexing with periodic scheduling to minimize busy waiting. In their setting, clients do not know the schedule.

To lengthen the operational lifetime of battery powered devices such as sensors, several power states are often maintained, such as receiving, transmitting, idle or standby. When devices change states in the course of normal operation, the power consumption often greatly varies from one state to another. Thus rebooting sensors may consume more energy than is saved by allowing sensors to go into sleep mode [58][48]. Consider a microserver, which normally remains in a low-power sleep mode. When a request arrives, it transfers to a higher-power state, determines whether it must process the request, and then returns to the low-power state. Platforms that provide sufficient resources to act as a microserver have high energy costs associated with transition, and that cost must be paid for every request [14]. Wowra [59] proposes the Self-Tuning Network Power Management, where the behavior of a network device adapts to access patterns in order to reduce energy consumption.

The same is true for individual components on a device which may be put into a dormant state when not receiving. For instance, the energy consumption of a small radio [2] in their dormant state is three orders of magnitude smaller than while it is receiving. However, the power required for a transition from the dormant state to the receiving state is roughly equal to the power drawn in the receiving state. Likewise, the power consumed by small disk drives, such as those employed in MP3 players [3], when writing received data exceeds the power consumed by the same disk when it is in standby by one order of magnitude. However, a transition from standby to writing is twice as expensive (in terms of power) as is writing itself.

We distinguish between the listening and reboot costs. We aim at minimizing them for each sensor. Thus, our goal is twofold: construct an efficient schedule that minimizes the obsoleteness of the data on the channel, and—simultaneously—minimize two diametrically opposed indicators: rebooting cost and the busy-waiting. This setting has not been considered before.

2.3 Model

2.3.1 Basic Model: Gap Driven Cost

Broadcast Channel: Pages are broadcast to sensors. The following are the assumptions of our model:

- $S_1, S_2, ..., S_n$ is a collection of *sensors.*
- $P_1, P_2, ..., P_n$ is a collection of equal-sized *info-pages* (tasks.) For simplicity, we use $A, B, C, ...$ instead of $P_1, P_2, P_3, ...$, respectively.
- Time is measured by discrete time slots; the first slot is numbered as $t = 1$; we assume that all sensors have updated pages at that time.
- A *cyclic slotted schedule* of info-pages is a sequence $P_{i_1}, P_{i_2}, ..., P_{i_T}$, where $i_j \in \{1, 2, ..., n\}$, and each page appears at least once. T is the *period* of the cyclic schedule; evidently: $T \geq n$.
- Delivery of page P_{i_k} happens during the time slot k.
- Define $\omega : \{1, \ldots, t\} \rightarrow \{1, \ldots, n\}$ such that: $\omega(x) = i$, meaning that a page for sensor S_i is broadcast at slot x.

Missions:

- Every mission is executed within a single time slot, by exactly one sensor. If a mission is executed at time slot t by a sensor S_i, then we say that S_i *operates at slot* t.

- Missions are executed perpetually; sensor S_i operates in an individual time-slot t. We assign to each sensor S_i a parameter p_i called *popularity* of this sensor's mission.

Execution of a mission is a customary practical term, which we model by operation of one sensor in one time slot. Every sensor (in general) is intended to operate infinitely many times (i.e., every mission, in general, is executed infinitely often.) The vector p specifies how different missions (sensor activations) are interleaved over time.

Popularity p is a feature in the model that allows us to attach a degree of flexibility and subjectivity to the calculation of costs. p_i is an instrument that allows us to quantify the relative importance of individual missions, as perceived by the owner of the system, depending on system properties that are not modeled for cost calculation. If such external concerns are absent, the most straightforward case of p_i = 1 may hold. Otherwise, values of p_i may be tuned to convey such perception of importance of individual missions relative to each other as is the most suitable to the situation at hand. Observe that the sum of all p_i may be greater than one – p_i are not probabilities.

Cost: Assume a mission is to be executed by S_i ($1 \leq i \leq n$) at time t, such that $1 \leq t \leq T$, where T is the period of the schedule cycle. Assume that the most recent page for S_i is sent at time t'. Then, we charge $g_i(t)$ to the cost, where g_i is a suitably defined *obsoleteness function* for sensor S_i. While it is a common assumption that $g_i(t)$ grows linearly with $t - t'$, other functions are plausible.

We assume a linear growth with a coefficient equal to 1:

$$g_i(t) = t - t' + 1$$

Hence, the *page obsoleteness cost* per slot, for all sensors of this schedule is equal to:

$$\mathcal{C}_g = \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i=1}^{n}(p_i \cdot g_i(t))\right) \tag{2.3.1}$$

After changing of order of summation:

$$\mathcal{C}_g = \sum_{i=1}^{n}\left(p_i \cdot \frac{\sum\limits_{t=1}^{T} g_i(t)}{T}\right) \tag{2.3.2}$$

In examples we usually use equation 2.3.2.

Example: Consider two sensors S_1 and S_2 whose popularities are $p_1 = 4/5$ and $p_2 = 1/5$ respectively. Page A is for sensor S_1 and page B is for sensor S_2.

Schedule	Period	Cost using formula (2.3.2)
$(AB)^*$	2	$\frac{4}{5}\cdot\frac{1+2}{2}+\frac{1}{5}\cdot\frac{1+2}{2}=\frac{3}{2}=1.5$
$(AAB)^*$	3	1.47
$(AAAB)^*$	4	1.5
$(AABB)^*$	4	1.75
$(AAAAB)^*$	5	1.56
$(AAABB)^*$	5	1.72

The schedule AAB emerges as the best among those considered. In fact, Ammar and Wong [11] show that AAB is optimal across all possible schedules, under $\mathcal{C}_g$.

In a non-regular schedule, where gaps between appearances of an individual page are not equal-sized, $\mathcal{C}_g$ is greater than in a regular schedule, where such gaps are all of equal size. Consider the schedule: $D' = ABAAAB$. By Equation (2.3.2), the schedule cost is: $\mathcal{C}_g(D') = 45/30 = 1.5$. Now consider another schedule: $D'' = AABAAB$. By Equation (2.3.2), the schedule cost is:

$\mathcal{C}_g(D'') = 44/30 \approx 1.47 < \mathcal{C}_g(D')$.

If the first B in D' is moved from the second place to the third, $\mathcal{C}_g$ decreases. This happens because the obsoleteness cost increases quadratically with gap size. Hence, two gaps of the same size are preferable to two gaps of unequal sizes, given that the sum of these two gaps is the same. Optimal schedules would favor uniform gaps between pages intended for the same sensor.

In general, A ought to be scheduled more frequently than B (since its popularity is higher.) However, this would lead to a relative increase in the size of B's gap. Thereby, gains attained by scheduling the more popular A more frequently would be offset by losses caused by unequal gaps. Ammar and Wong [11] show that in an optimal schedule, a gap associated with a sensor is proportional to the square root of the sensor's popularity. Hence, ratio $\sqrt{4}$ to $\sqrt{1}$ is optimal.

Blank slots are not interesting in the basic model, since every schedule with a blank slot is trivially less efficient than any schedule where that blank slot is substituted by any page. This is because the entire cost is given by a data cost, which cannot be improved by omitting any page. Once sensor related costs are taken into account, empty slots may become profitable.

2.3.2 Sensor Types and Modified Model

In the basic model, we have considered only sensors that have unlimited power supply. However, sensors often operate on a battery, and such a battery has a limited life-time. Therefore, we have to consider different types of sensors.

1. Sensors that are continuously on (active);

2. Sensors that can be powered OFF (to save power), and switched back ON at no cost;

3. Sensors that can be powered OFF (to save power), but the cost to switch them ON is not equal to zero.

An active sensor is continuously listening to the broadcast channel, even if it is not executing any missions. This type of sensor incurs another kind of cost called a *listening* cost. The longer the sensor is active, the greater the *listening* cost. Hence, it makes sense to power OFF sensors whenever they do not execute their missions. This strategy is good for those sensors whose switch-ON cost is equal to zero. However, for those sensors whose switch-ON cost is not equal to zero, such strategy would incur yet another cost called *switch* cost. Our goal is to minimize the *listening* and the *switch* costs as well.

Listening

Recall that sensors in the basic model may be left to operate perpetually, since their power supply is unlimited by assumption. However, if we are to take the battery life into account, it appears advisable to switch the battery OFF when a sensor is idle. In every time-slot when the sensor is not idle, some listening cost is incurred. Therefore, in each time slot, we switch ON just the sensor which is executing its mission.

It may seem impossible for a sensor to have "lookahead" into its future missions in those cases where missions are initiated by the system (rather than by sensors themselves, or pre-assigned in some static scheme.) However, techniques have been developed to allow radios to enter a sleep state from which they awaken without a pre-determined schedule when they are required to received data. These techniques result in nodes operating in at least three different power states: sleep, idle (i.e., monitoring), and receiving. Transitions between these states incur a switching cost.

Two approaches to solving the lookahead problem have become popular; they have different costs. In the first approach, Rabaey *et al.*[47, 46] show how to build circuitry to detect an incoming signal on the channel, before it reaches the node (radio) which it powers up. In the second approach, Reason and Rabaey [50] impose a wake-up regimen (called duty-cycle) on the nodes, so that they can test the channel for a signal. This requires a transmitter to sends a relatively long preamble, prior to transmitting the actual data. If a node awakens during the preamble, it remains ON to receive data. The length of the preamble and duty cycles may be tuned, so as to decrease the likelihood of missing data.

Both of these approaches incur a cost to determine when to wake up a node and to switch the node from the sleep mode. The switching cost does not depend on the method employed to determine when to wake up, but depends on the type (electronic properties) of the node. The first approach incurs some cost to maintain the wake-up circuitry. The second approach incurs costs of periodic wake-up even when no data is present.

Studies [50] have shown the cost of a preamble-sampling system to be about twice as high as that of a system in which a node "knows" when to wake-up. The act of switching a radio ON requires as much power as receiving data.

In this paper we remain agnostic to the method of activating a radio from sleep state and switching it ON. We model this cost as a composite value.

Let λ_j be the listening cost incurred by an individual sensor S_j whenever S_j is up. We formalize the *listening* cost of a schedule as:

$$\mathcal{C}_\lambda = \frac{1}{T}\sum_{t=1}^{T} \lambda_{\omega(t)} \tag{2.3.3}$$

Putting these two costs (obsoleteness and listening) together, we obtain the total cost per slot: $\mathcal{C} = \mathcal{C}_g + \mathcal{C}_\lambda$ or

$$\mathcal{C} = \frac{1}{T}\sum_{t=1}^{T}\left(\lambda_{\omega(t)} + \sum_{i=1}^{n}(p_i \cdot g_i(t))\right) \tag{2.3.4}$$

Example: Consider our schedules, already compared under the basic model, with two sensors, S_1 and S_2 whose popularities are $p_1 = 4/5$ and $p_2 = 1/5$ respectively. Assume that the listening cost is independent of time; let λ_A and λ_B be the listening costs of sensors S_1 and S_2, respectively.

Case (1) $\lambda_A = \lambda_B$: Schedule AAB is optimal.

Case (2) $\lambda_A > \lambda_B$: Consider the schedules: $D_1 : AAB$ and $D_2 : AB$.

$$\mathcal{C}(D_1) = 1.47 + \frac{2}{3}\lambda_A + \frac{1}{3}\lambda_B$$

$$\mathcal{C}(D_2) = 1.5 + \frac{1}{2}\lambda_A + \frac{1}{2}\lambda_B$$

Since for $\lambda_A - \lambda_B > 0.18$

$$1.47 + \frac{2}{3}\lambda_A + \frac{1}{3}\lambda_B > 1.5 + \frac{1}{2}\lambda_A + \frac{1}{2}\lambda_B$$

it follows that $\mathcal{C}(D_1) > \mathcal{C}(D_2)$. Hence, D_2 is better than D_1. Thus, if sensor S_1 operates on a more costly battery than sensor S_2, it is no longer optimal to schedule as in the model with obsoleteness cost only. Now, sensor S_1 should be allowed more obsolete data—less frequent page delivery.

Case (3) $\lambda_A < \lambda_B$: The two schedules are: $D_1 : AAB$ and $D_3 : AAAB$.

$$\mathcal{C}(D_1) = 1.47 + \frac{2}{3}\lambda_A + \frac{1}{3}\lambda_B$$

$$\mathcal{C}(D_3) = 1.5 + \frac{3}{4}\lambda_A + \frac{1}{4}\lambda_B$$

Since for $\lambda_B - \lambda_A > 0.36$

$$1.47 + \frac{2}{3}\lambda_A + \frac{1}{3}\lambda_B > 1.5 + \frac{3}{4}\lambda_A + \frac{1}{4}\lambda_B$$

it follows that $\mathcal{C}(D_1) > \mathcal{C}(D_3)$. Hence, D_3 is better than D_1. Thus, if sensor S_2 operates on a more costly battery than sensor S_1, it is no longer optimal to schedule as in the basic model. Now, sensor S_2 should be allowed more obsolete data—less frequent page delivery.

Gaps and Listening, and Non-Linear Programming

Suppose sensor i is operating at time slot t' and is not operating in time slots $t'+1, ..., t'+x-1$. Then, the total obsoleteness cost incurred by sensor i in the x time slots $t', t'+1, ..., t'+x-1$ is given by:

$$\mathcal{C}_i = \sum_{j=0}^{x-1} (p_i \cdot (j+1)) = \frac{p_i}{2}(x+1)x$$

Suppose the set of variables $\{\tau_1, \tau_2, ..., \tau_n\}$ are periods of pages in the schedule. The lower bound on the long-term average cost of an optimal schedule is:

$$\frac{1}{2}\sum_{i=1}^{n} (p_i \cdot (\tau_i + 1)) + \sum_{i=1}^{n} \frac{\lambda_i}{\tau_i}$$

The non-linear mathematical program lower bound in the combined model with gaps and listening is:

The solution to the mathematical program gives a fractional lower bound, since periods may cause overlap of broadcast data. Such is the case, for instance, when $\tau_1 = 2$ and $\tau_2 = 3$, saying that page A needs to appear every 2 slots and page B every 3 slots.

Table 2.1: Mathematical program for LB of gaps and listening model.

$$\begin{aligned} \min \ & \sum_{i=1}^{n} \left[\frac{\tau_i + 1}{2} p_i + \frac{\lambda_i}{\tau_i} \right] \\ \text{s.t.} \ & \sum_{i=1}^{n} \left(\frac{1}{\tau_i} \right) \leq 1 && (2.3.5) \\ & \tau_i \geq 1 \qquad \text{for all } 1 \leq i \leq n && (2.3.6) \end{aligned}$$

Gaps, Listening, and Switch

In the absence of any switch-ON cost, it is preferable not to have a sensor listen unless it is operating. However, if a switch costs, it may be better to let the sensor listen during an idle time-slot than switch ON at the beginning of the next slot. Now, our goal is to schedule not only pages but also sensor switches. We start out by investigating scheduling of pages when provided with a fixed, given schedule of sensor switches.

Let σ_j be the switch cost incurred by an individual sensor S_j whenever S_j is being switched off. We define the *switch pattern* $\psi(t)$ as follows:

$$\psi(t) = \begin{cases} 0 & \text{if sensor } \omega(t) \text{ is not switched off;} \\ 1 & \text{if sensor } \omega(t) \text{ is switched off.} \end{cases}$$

Observe that the value of the reboot cost ought to be assigned so as to model all relevant aspects of the realistic reboot, and the time slot ought to be allowed to be long enough to accommodate a reboot/shutdown. Also, note the implicit assumption that a sensor may be switched OFF only at (the end of) a time slot when it operates. Once we have to pay a switch cost, it makes no sense to wait any longer, since waiting only increases the listening cost.

Thus, the total switch cost incurred by the entire schedule is:

$$\mathcal{C}_\sigma = \frac{1}{T}\sum_{t=1}^{T}\left(\sigma_{\omega(t)} \cdot \psi(t)\right)$$

In the case when $\psi(t) = 0$ for all t, no switch cost is charged to any sensor. However, all sensors are subject to the listening cost continuously. On the other hand, when $\psi(t) = 1$ for all t, sensors are switched OFF immediately at the end of their slots, each time contributing to the switch cost but incurring no unnecessary listening cost.

In a more realistic case, we have a situation when $\psi(t)$ is neither all 0's nor 1's. For a sensor S_i, a *listening interval of length k starting at m* is a sequence of contiguous slots starting at slot m and ending at slot $m + k - 1$, such that all of the following hold:

$\omega(m) = \omega(m + k - 1) = i$
$\psi(m + k - 1) = 1$
for all j such that $m \leq j < m{+}k{-}1, \psi(j){=}0$ if $\omega(j){=}i$

For a sensor S_i, let the *listening length* $\gamma(i)$ be the sum of lengths of all listening intervals associated with this sensor.

Thus, the listening cost becomes:

$$\mathcal{C}_\lambda = \frac{1}{T}\sum_{i=1}^{n}\left(\gamma(i) \cdot \lambda_i\right)$$

Now, the total schedule cost is the sum of all these costs:

$$\mathcal{C} = \mathcal{C}_g{+}\mathcal{C}_\lambda{+}\mathcal{C}_\sigma$$

$$\mathcal{C} = \frac{1}{T}\left[\sum_{t=1}^{T}\sum_{i=1}^{n}(p_i{\cdot}g_i(t)){+}\sum_{i=1}^{n}(\gamma(i){\cdot}\lambda_i){+}\sum_{t=1}^{T}\left(\sigma_{\omega(t)}{\cdot}\psi(t)\right)\right]$$

Example: Consider our schedules, already compared under the basic and listening model, with two sensors, S_1 and S_2 whose popularities are $p_1 = 4/5$ and $p_2 = 1/5$ respectively.

Case (1) $\lambda_A > \lambda_B$: Two schedules: $D_1 : AAB$ and $D_2 : AB$

Recall that D_2 is better than D_1 whenever $\lambda_A - \lambda_B > 0.18$. Now, assume that $\lambda_A = \lambda_B + 0.2$, and the switch pattern turns a sensor OFF whenever there is no page update for it within the next two time slots.

D_1 :				D_2 :		
slot t	1	2	3		1	2
page	A	A	B		A	B
$\psi(t)$	0	0	1		0	0

$$\mathcal{C}(D_1) = 1.47 + \frac{3}{3}\lambda_A + \frac{1}{3}\lambda_B + 0 \cdot \sigma_A + \frac{1}{3}\sigma_B$$
$$\mathcal{C}(D_2) = 1.5 + \frac{2}{2}\lambda_A + \frac{2}{2}\lambda_B + 0 \cdot \sigma_A + 0 \cdot \sigma_B$$

When is $\mathcal{C}(D_1) < \mathcal{C}(D_2)$?

Substitute $\lambda_A = \lambda_B + 0.2$ into $\mathcal{C}(D_1)$ and $\mathcal{C}(D_2)$ to get:

$$1.47 + \frac{3}{3}(\lambda_B + 0.2) + \frac{1}{3}\lambda_B + \frac{1}{3}\sigma_B < 1.5 + 2\lambda_B + 0.2$$

Equivalently:

$$\sigma_B + 0.09 < \lambda_B$$

Hence, under the switch cost, D_1 is better than D_2. Thus, if switching S_2 OFF is not that costly in comparison to its operating cost, D_1 may be better than D_2.

Case (2) $\lambda_A < \lambda_B$: Two schedules: D_1 and D_3, where $D_1 = AAB$ and $D_1 = AAAB$. Recall that D_3 is better than D_1 whenever $\lambda_B - \lambda_A > 0.36$, Now, assume that:

$$\lambda_B = \lambda_A + 0.4$$

Under this condition, with no switch cost, D_3 is better. Assume a switch pattern that turns OFF a sensor whenever there is no page update for it within the next three time slots.

D_1 :				D_3 :				
slot t	1	2	3		1	2	3	4
page	A	A	B		A	A	A	B
$\psi(t)$	0	0	0		0	0	0	1

$$\mathcal{C}(D_1) = 1.47 + \frac{3}{3}\lambda_A + \frac{3}{3}\lambda_B + 0 \cdot \sigma_A + 0 \cdot \sigma_B$$
$$\mathcal{C}(D_3) = 1.5 + \frac{4}{4}\lambda_A + \frac{1}{4}\lambda_B + 0 \cdot \sigma_A + \frac{1}{4}\sigma_B$$

When is $\mathcal{C}(D_1) < \mathcal{C}(D_3)$?

Substitute $\lambda_A = \lambda_B - 0.4$ into $\mathcal{C}(D_1)$ and $\mathcal{C}(D_1)$ to get:

$$1.47 + (\lambda_B - 0.4) + \lambda_B < 1.5 + \frac{4}{4}(\lambda_B - 0.4) + \frac{1}{4}\lambda_B + \frac{1}{4}\sigma_B$$

Equivalently:

$$\sigma_B > 3\lambda_B - 0.12$$

Hence, under the switch cost, D_1 is better than D_3. Thus, D_1 is better than D_3 if switching S_2 OFF costs three times as much as letting it operate for a slot, assuming S_2 is left listening.

Based on the mathematical model, we have implemented a simulation environment to evaluate and compare different schedules. We consider two cases in our simulation. In the first, there is only one sensor and the goal is to find the balance between the obsoleteness cost and the sensor cost. In the second, there are many sensors and the goal is to implement a reasonable greedy solution that would produce an efficient schedule. In the next section, we present an one-sensor setting problem and a non-linear program that will find a lower bound for the setting of many sensors.

2.4 One Sensor

The one-sensor case is interesting in its own right. Our objective is to schedule updates for one sensor optimally.

In the absence of sensor costs, it would be optimal to schedule the data page (denoted by A) in each slot. If the only sensor cost is the listening cost while switching ON and OFF is free, then one can find a parameter x such that A followed by $(x-1)$ blank slots (denoted by b) is the optimal schedule. The schedule looks like $Abb...bAbb..bA...$. In a sense, we trade obsoleteness cost with listening cost (see [17]).

Once a non-zero switching cost is admitted, it is not immediately clear whether a schedule of type $Abb...$ is better than a schedule of type $AAbbb...$. In other words, the question is whether the best schedule comprises A followed by blank slots, or more than one A followed by blank slots. The assumption is that the switching OFF happens exactly after the last A in any consecutive sequence of occurrences of A.

To gain insight, we simulated a broadcast system and varied the number of occurrences of A and the number of occurrences of b. The length of the schedule T at most 100. The simulation also varies values of popularity p and sensor's listening and switch costs per slot, λ and σ. The simulation results indicate that one A followed by zero or more b's is almost always better than multiple A's. However, for some values of p, σ, and λ, the simulation produces solutions where it is better to schedule x occurrences of A followed by $(T-x)$ occurrences of b. Yet, if we employ $T+y$ instead of T (all else equal), the number of A's that leads to a better schedule is $x+y$, followed by the same number of b's, i.e., $(T-x)$. In other words, the simulation indicates that in this scenario but in an infinite schedule, it would be preferable not to switch OFF, but to continue scheduling A. Thus, in this case a better schedule is to continuously scheduling A's, except for the suffix.

With this insight we now proceed to prove our result for infinite one-sensor schedules. First, we show that the infinite optimal schedule must be cyclic, of type $A^x b^y$, for some $x \geq 1$ and $y \geq 0$, and then we show that $x = 1$.

Lemma 2.4.1. *Given is a sensor in an infinite schedule. Assume that there is a switch* OFF *after every last A before blanks. Otherwise, since $n = 1$, keep broadcasting A. Then, there exist integers τ, x, y such that a schedule of the form $(A^x b^y)^*$ is optimal for this sensor, where $x + y = \tau$, $x \geq 1$ and $y \geq 0$. In other words, there exists an optimal periodic schedule with a period τ.*

Proof. Recall that this schedule is infinite. Partition it into subsequences as follows. Scan the infinite schedule from left to right. As long as blanks last, form subsequences out of maximal contiguous runs of the form $A^m b^\ell$. Once there are no more blanks (if ever), each A is a subsequence on its own. (For instance, a schedule *AAbbbbAAAAAbbbbbAbbA*... would be partitioned into subsequences *AAbbbb*, *AAAAAbbbbb*, *Abb*, *A*....) Calculate the cost per slot of each subsequence. Let x and y be such that the cost of the subsequence $A^x b^y$ is the minimum among all the subsequence costs. Substitute every subsequence whose cost is higher than minimum with the subsequence $A^x b^y$. The cost per slot decreases as a result of the substitution, and—by definition—the cost of the entire schedule also decreases. □

Observe that either the schedule has infinitely many blanks or none, depending on the structure of the subsequence for which the minimum cost occurs (which becomes the only sequence in the optimal schedule.) If the schedule has blanks, then they come structured as is stated in Lemma 2.4.2. Theorem 2.4.3 states the conditions which determine whether the schedule has blanks or not.

Lemma 2.4.2. *For any integers $x \geq 1$ and $y \geq 0$, and a periodic schedule $s = (A^x b^y)^*$*

such that $\mathcal{C}(A^x b^y) \leq \lambda + p$, the cost of the periodic schedule $(Ab^y)^*$ does not exceed the cost of s.

Proof. We have to show that the cost per slot of a subsequence Ab^y is less than or equal to the cost per slot of a subsequence $A^x b^y$, provided that cost per slot for $\mathcal{C}(A^x b^y) \leq \lambda + p$. In short, we need to prove that:

$$\mathcal{C}(Ab^y) \leq \mathcal{C}(A^x b^y)$$

where:

$$\mathcal{C}(Ab^y) = \frac{\sigma + \lambda}{1 + y} + \frac{(y^2 + 3y + 2)p}{2(1 + y)}$$

$$\mathcal{C}(A^x b^y) = \frac{\sigma + x\lambda}{x + y} + \frac{(y^2 + 3y + 2x)p}{2(x + y)}$$

If $x = 1$, the claim is evidently true, since equality holds. It remains to prove the claim assuming that $x \geq 2$. By assumption, $\mathcal{C}(A^x b^y) \leq \lambda + p$, meaning:

$$\frac{\sigma + x\lambda}{x + y} + \frac{(y^2 + 3y + 2x)p}{2(x + y)} \leq \lambda + p$$

This implies:

$$y\lambda - \frac{(y^2 + y)p}{2} - \sigma \geq 0$$

Since $\frac{(x-1)}{(x+y)(1+y)} > 0$ for any $x \geq 2$, we can multiply by it to get:

$$\frac{(x - 1)(2y\lambda - (y^2 + y)p - 2\sigma)}{2(x + y)(1 + y)} \geq 0$$

This implies that:

$$\frac{2xy\lambda - 2y\lambda - (y^2 x + yx - y^2 - y)p + (2\sigma - 2x\sigma)}{2(x + y)(1 + y)} \geq 0$$

After straightforward transformations:

$$\left(\frac{\sigma+x\lambda}{(x+y)}+\frac{(y^2+3y+2x)p}{2(x+y)}\right)-\left(\frac{\sigma+\lambda}{(1+y)}+\frac{(y^2+3y+2)p}{2(1+y)}\right)\geq 0$$

which is:

$$\mathcal{C}(A^x b^y)-\mathcal{C}(Ab^y)\geq 0$$

and also:

$$\mathcal{C}(s)-\mathcal{C}((Ab^y)^*)\geq 0$$

□

Theorem 2.4.3. *An optimal infinite schedule for one sensor is periodic and:*

1. *either is of the form $Ab^{\tau-1}$ with period $\tau=\left\lceil\sqrt{\frac{2(\lambda+\sigma)}{p}}\right\rceil$ or $\tau=\left\lfloor\sqrt{\frac{2(\lambda+\sigma)}{p}}\right\rfloor$ (whichever minimizes schedule cost per slot) in which case a switch* OFF *occurs after A;*
2. *or is of the form A^* and there is no switch* OFF.

Proof. Two cases are possible.

(1) it is optimal to switch OFF after A;

(2) it is not optimal to switch OFF after A.

According to Lemma 2.4.2, in the first case, a periodic schedule of the form $Ab^{\tau-1}$ is optimal, for some period $\tau\geq 1$. Otherwise, in the second case, we have an infinite sequence A^*.

Now, we solve for the period τ of our optimal schedule.

If we switch, the cost of the schedule per slot is

$$\mathcal{C}_1=\frac{x+1}{2}\cdot p+\frac{\lambda+\sigma}{x}$$

If we do not switch, the cost of the schedule per slot is

$$\mathcal{C}_2 = \frac{x+1}{2} \cdot p + \lambda$$

We need to find x that minimizes the cost among the two cases. The minimum of $\mathcal{C}_2$ occurs when $x = 1$ and in this case:

$$\mathcal{C}_2 = \lambda + p$$

To find the value of x that minimizes $\mathcal{C}_1$, find a derivative of $\mathcal{C}_1$ with respect to x, set it to 0, and solve for x. The minimum occurs for

$$x_0 = \sqrt{\frac{2\,(\lambda + \sigma)}{p}}$$

Here, x_0 is not an integer. Let τ be either $\lceil x_0 \rceil$ or $\lfloor x_0 \rfloor$, (whichever minimizes schedule cost per slot).

For $x = \tau$, calculate $\mathcal{C}_1$. If $\mathcal{C}_2 < \mathcal{C}_1$, then the schedule is of the form A^*. Otherwise, the schedule is of the form $Ab^{\tau-1}$. $\square$

In summary, the optimal infinite schedule for a single sensor is either a single page followed by blanks, or the page repeated in all slots, depending on the relationship among the cost values σ, λ, p.

2.5 Lower Bound

In this section we derive a lower bound on the optimal solution cost for many sensors. Our solution for one sensor is employed to derive a mathematical program to find a fractional lower bound on the cost per slot of schedules with n sensors.

Table 2.2: Mathematical program for LB of the model.

$$\begin{aligned}
\min \; & \sum_{i=1}^{n} \left[\frac{\tau_i + 1}{2} p_i + \min\left(\frac{\lambda_i + \sigma_i}{\tau_i}, \lambda_i \right) \right] \\
\text{s.t.} \; & \sum_{i=1}^{n} \left(\frac{1}{\tau_i} \right) \le 1 && && (2.5.1) \\
& \tau_i \ge 1 && \text{for all } 1 \le i \le n && (2.5.2)
\end{aligned}$$

Let $\tau^* = (\tau_1, \tau_2, ..., \tau_n)$ be the solution of the following program.

The expression $\min\left(\frac{\lambda_i+\sigma_i}{\tau_i}, \lambda_i\right)$ is equal to λ_i exactly when sensor i is not switched OFF, and is equal to $\frac{\lambda_i+\sigma_i}{\tau_i}$ when sensor i is switched OFF.

The solution τ^* gives a fractional lower bound, which need not be attainable. For instance, when $\tau^* = (\tau_1, \tau_2)$ where $\tau_1 = 2$ and $\tau_2 = 3$, one cannot schedule according to τ^*, or else there is a conflict at slot number $6t$, for all $t \ge 1$.

Consider the above non-linear mathematical program to find a fractional lower bound on the cost per slot of schedules with n sensors.

Theorem 2.5.1. *An optimal solution to program in table (2.2) is given by*

$$\tau_i = \max\left(1, \sqrt{\frac{2(\lambda^* + \lambda_i + \sigma_i)}{p_i}}\right) \quad \textit{in case of a switch.}$$

$$\textit{or} \;\; \tau_i = \max\left(1, \sqrt{\frac{2\lambda^*}{p_i}}\right) \quad \textit{in case of no switch.}$$

$(i = 1...n)$, *where* $\lambda^* \ge 0$. *If* $\lambda^* > 0$ *then* $\sum_{i=1}^{n}\left(\frac{1}{\tau_i}\right) = 1$.

Proof. We obtain the following non-linear program by applying Lagrangian relaxation to program in table (2.2).

Minimize:

$$\sum_{i=1}^{n} \frac{(1+\tau_i)}{2} p_i + \sum_{i=1}^{n} \min\left(\lambda_i, \frac{\lambda_i + \sigma_i}{\tau_i}\right) - \lambda\left(1 - \sum_{i=1}^{n} \frac{1}{\tau_i}\right) - \sum_{i=1}^{n} \mu_i(\tau_i - 1) \tag{2.5.3}$$

Take partial derivatives with respect to τ_i. In a case when there is no switch, $\min\left(\lambda_i, \frac{\lambda_i+\sigma_i}{\tau_i}\right) = \lambda_i$. The partial derivatives with respect to τ_i are $\frac{p_i}{2} - \frac{\lambda}{\tau_i^2} - \mu_i$. Setting them to zero and solving for τ_i gives $\tau_i = \sqrt{\frac{2\lambda}{p_i - 2\mu_i}}$. In a case when there is a switch, $\min\left(\lambda_i, \frac{\lambda_i+\sigma_i}{\tau_i}\right) = \frac{\lambda_i+\sigma_i}{\tau_i}$. The partial derivatives with respect to τ_i are $\frac{p_i}{2} - \frac{\lambda_i+\sigma_i}{\tau_i^2} - \frac{\lambda}{\tau_i^2} - \mu_i$. Setting them to zero and solving for τ_i gives $\tau_i = \sqrt{\frac{2(\lambda+\lambda_i+\sigma_i)}{p_i-2\mu_i}}$.
For any fixed $\lambda \geq 0$ and $\mu_i \geq 0$ the optimal value of the non-linear program (2.5.3) is a lower bound on the optimal value of the non-linear program in table (2.2), and an optimal solution of (2.5.3) is given by $\tau_i = \sqrt{\frac{2\lambda}{p_i-2\mu_i}}$ when there is no switch or $\tau_i = \sqrt{\frac{2(\lambda+\lambda_i+\sigma_i)}{p_i-2\mu_i}}$ when there is a switch, provided that $p_i - 2\mu_i > 0$.

Now, we fix $\lambda = 0$ and $\mu_i = 0$ and set $\tau_i = max(1, 0) = 1$ (in a case of no switch) or $\tau_i = max(1, \sqrt{\frac{2(\lambda_i+\sigma_i)}{p_i}})$ (in a case of a switch). This assignment may not be feasible for program in table (2.2) since it may be the case that $\sum_{i=1}^{n} \frac{1}{\tau_i} > 1$.

In this case increase λ until the latter constraint is satisfied, and define $\lambda^* > 0$ to be the new value λ for which

$$\sum_{i=1}^{n} \frac{1}{\tau_i} = \sum_{i=1}^{n} \left[\min\left(1, \sqrt{\frac{p_i}{2(\lambda+\lambda_i+\sigma_i)}}\right) \text{ or } \min\left(1, \sqrt{\frac{p_i}{2\lambda}}\right)\right] = 1 \tag{2.5.4}$$

In the case $\sum_{i=1}^{n} \frac{1}{\tau_i} \leq 1$ we set $\lambda^* = 0$.

For all i such that $\sqrt{\frac{2(\lambda+\lambda_i+\sigma_i)}{p_i}}$ (for a switch case) $\left[\text{or } \sqrt{\frac{2\lambda}{p_i}} \text{ (for a no switch case)}\right]$ is strictly less than 1, set μ_i to the positive value for which $\sqrt{\frac{2(\lambda+\lambda_i+\sigma_i)}{p_i-2\mu_i}} = 1$ (for a switch case)
$\left[\text{or } \sqrt{\frac{2\lambda}{p_i-2\mu_i}} = 1 \text{ (for a no switch case)}\right]$.

Note that for this choice of λ and $\mu_i (1 \leq i \leq n)$ we have that $\sqrt{\frac{2(\lambda+\lambda_i+\sigma_i)}{p_i-2\mu_i}} = \max\left(1, \sqrt{\frac{2(\lambda+\lambda_i+\sigma_i)}{p_i}}\right)$ $\left[\text{or } \sqrt{\frac{2\lambda}{p_i-2\mu_i}} = \max\left(1, \sqrt{\frac{2\lambda}{p_i}}\right)\right]$ for $1 \leq i \leq n$ and for the assignment

$$\tau_i = \max\left(1, \sqrt{\frac{2(\lambda^* + \lambda_i + \sigma_i)}{p_i}}\right) \quad \text{or}$$

$$\tau_i = \max\left(1, \sqrt{\frac{2\lambda^*}{p_i}}\right)$$

we have $\mu_i(\tau_i - 1) = 0$ and $\lambda\left(1 - \sum_{i=1}^{n} \frac{1}{\tau_i}\right) = 0$. This implies that

$$\tau_i = \max\left(1, \sqrt{\frac{2(\lambda^* + \lambda_i + \sigma_i)}{p_i}}\right) \quad \text{or}$$

$$\tau_i = \max\left(1, \sqrt{\frac{2\lambda^*}{p_i}}\right)$$

is optimal solution for both equation (2.5.3) and equation in table (2.2). $\square$

Note that the objective function in equation in table (2.2) for each i is not convex but two pieces of convex function. There is a threshold value that separates when there is no switch and when there is a switch. Let a function $\mathcal{C}_{i,no-switch}(\tau_i)$ be the cost associated with sensor i when $\min\left(\lambda_i, \frac{\lambda_i+\sigma_i}{\tau_i}\right) = \lambda_i$ (i.e. there is no switch) and let a function $\mathcal{C}_{i,switch}(\tau_i)$ be the cost associated with sensor i when $\min\left(\lambda_i, \frac{\lambda_i+\sigma_i}{\tau_i}\right) = \frac{\lambda_i+\sigma_i}{\tau_i}$ (there is a switch). The threshold τ_i' is when $\mathcal{C}_{i,no-switch} = \mathcal{C}_{i,switch}$. Observe that $\mathcal{C}_{i,no-switch}$ is a monotonically increasing linear function and $\mathcal{C}_{i,switch}$ is a convex function that has a single minimum value. We observe further that the tangent at any τ_i on a switch curve is at most the tangent at any τ_i on a no-switch curve. Now, the goal is to minimize this piecewise convex function. We minimize each $\mathcal{C}_{i,no-switch}$ and $\mathcal{C}_{i,switch}$. Let these minimum values be $\mathcal{C}^m_{i,no-switch}$ and $\mathcal{C}^m_{i,switch}$ respectively. We consider no switch to begin if $\mathcal{C}^m_{i,no-switch} < \mathcal{C}^m_{i,switch}$, otherwise we consider a switch. Observe that these values are when Lagrange multipliers λ and μ_i are set to zero. When finding a global minimum for the equation in table 2.2 this assignment may not be feasible and so we increase the λ. Note that if we are already on the switch curve (i.e. $\mathcal{C}_{i,switch}(\tau_i)$) when increasing λ, there will be no incentive in going back to a no-switch curve, since τ_i for a switch curve is bigger than τ_i', whereas τ_i on a no-switch curve is smaller than τ_i', and we prefer larger τ_i in order to satisfy the constraint $\sum_{i=1}^{n} \frac{1}{\tau_i} \leq 1$. In addition the tangent of a switch curve is at most the tangent of a no-switch curve at any point. Thus, once we are on a switch curve, it is better to stay on a switch curve. However, if we are on a no-switch curve (i.e. $\mathcal{C}_{i,no-switch}(\tau_i)$) when increasing λ, if we increase λ enough to have $\sqrt{\frac{2\lambda}{p_i}} = \mathcal{C}^m_{i,switch}$, then we change to be on a switch curve (i.e. $\mathcal{C}_{i,switch}(\tau_i)$), reset the λ back to zero and start over increasing λ until the constraint $\sum_{i=1}^{n} \frac{1}{\tau_i} \leq 1$ is satisfied. We will have at most n curve changes when finding a global minimum for the Lagrangian equation (2.5.3), rather than testing all 2^n different possibilities.

2.6 Many Sensors

Multiple sensors compete for slots. Near-optimal solutions are known on assumption that rebooting incurs no cost. One of these solutions is based on a greedy rule [17]. This rule selects the next data page minimizing the total cost among all possible selections. We adapt to our model these greedy heuristics, developed in [17] for the model without rebooting cost, and compare them in a simulation environment. Following is a selection from our experiments.

2.6.1 Heuristics

```
Let $w_i(t)$ be the number of slots during which page $P_i$ is not scheduled before timeslot $t$
$w_i(0) = 0, \forall i \ (1 \leq i \leq n)$
Let $q_i = \frac{\sqrt{p_i}}{\sum_n^{j=1} \sqrt{p_j}}$ (*for Original Greedy*)
  or $q_i = \frac{\sqrt{p_i/\lambda_i}}{\sum_n^{j=1} \sqrt{p_i/\lambda_i}}$ (*for Modified Greedy*)
Let $\mu_i = \frac{\sigma_i+\lambda_i}{\lambda_i}$
**for** $t = 1$ **to** $T$ **do**
  **for** $i = 1$ **to** $n$ **do**
    $r_i \leftarrow (1 + w_i(t))q_i$
    $w_i \leftarrow w_i + 1$
  **end for**
  Let sensor $m$ be the one for which $r_i$ is maximal
  Schedule page $P_m$
  $w_m \leftarrow 0$
**end for**
Schedule switches for sensor $S_i$ if there is no data update within $\mu_i$ timeslots
```

Algorithm 1: Greedy Algorithms

Strategy 1: Ignore all sensor costs (listening and switch) and establish the best solution based on obsoleteness cost only.

Strategy 2: Ignore only the switch cost and establish the best solution based on obsoleteness cost and listening cost.

In both strategies, the greedy rule selects a task as follows.

Original and Modified Greedy Rules:

Let $w_i(t)$ be the number of slots during which task P_i is not scheduled before slot t; assume

$w_i(0) = 0$ for all i. Let q_i be a suitably chosen function (depending on the rule itself—original or modified) such that $\sum_{i=1}^{n} q_i = 1$; q_i is referred to as a *normalized popularity* of mission execution for sensor S_i in a time slot.

Schedule P_i at t such that $r_i = (1 + w_i(t))q_i$ is maximal.

Example for *Strategy 1:*

Assume:

$$\{p_1, p_2, p_3\} = \left\{\frac{9}{14}, \frac{5}{14}, \frac{1}{14}\right\}$$

and q is computed as:

$$\{q_1, q_2, q_3\} = \left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right\}$$

The first three runs of the greedy selection are as follows:

1. $\{r_1, r_2, r_3\} = \left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right\} \Rightarrow$ Schedule is P_1

2. $\{r_1, r_2, r_3\} = \left\{\frac{1}{2}, \frac{2}{3}, \frac{1}{3}\right\} \Rightarrow$ Schedule is P_1P_2

3. $\{r_1, r_2, r_3\} = \left\{1, \frac{1}{3}, \frac{1}{2}\right\} \Rightarrow$ Schedule is $P_1P_2P_1$

In *Strategy 2,* a different vector of normalized probabilities q is calculated, and the corresponding schedule is constructed.

For sensor S_i, define a *threshold:*

$$\mu_i = (\sigma_i + \lambda_i)/\lambda_i$$

Sensor S_i is switched OFF exactly when there is no data for S_i within a time interval of size μ_i.

Another greedy rule is based on the Golden Ratio [17]. *Golden-Ratio Rule:* Compute q_i. Then, plot the vector q along a line of unit length, so that to each q_i corresponds a line segment of length q_i. Close the endpoints of the line into a circle. Label the segment that corresponds to q_i as P_i, for all i. Now, let $\hat{\phi} = 0.61803399$. Mark a point on the circle that is located at a distance of exactly $\hat{\phi}$ from a designated point 0. Read off the label where this point is, and schedule that task. Continue recursively from this point, traversing the circle in steps of length $\hat{\phi}$, scheduling tasks named by the labels encountered. Schedule the switches according to the threshold μ.

2.6.2 Simulation Stage

Experiments have been carried out for the *round-robin* schedule, where page P_i is broadcast in every n^{th} slot, for the Original and Modified Greedy Rules, and for the Golden-Ratio Rule. The popularities of mission executions are assumed to follow the Zipf distribution with parameter α, for varying values of α. As α increases, the difference in popularity of the missions widens.

Since our model is original, these are the first performance studies in this model. Hence, in the absence of a more advanced base-line, our simulation studies compare our solutions with the round-robin scheduling only.

Based on popularities p_i, q_i are computed for:

Strategy 1: $q_i = \sqrt{p_i}/(\sum_n^{j=1} \sqrt{p_j})$

and for:

Strategy 2: $q_i = \sqrt{p_i/\lambda_i}/(\sum_n^{j=1} \sqrt{p_i/\lambda_i})$.

Time is allocated to tasks according to q_i. Precisely, for any finite prefix of the schedule, a fraction of q_i slots is dedicated to page P_i.

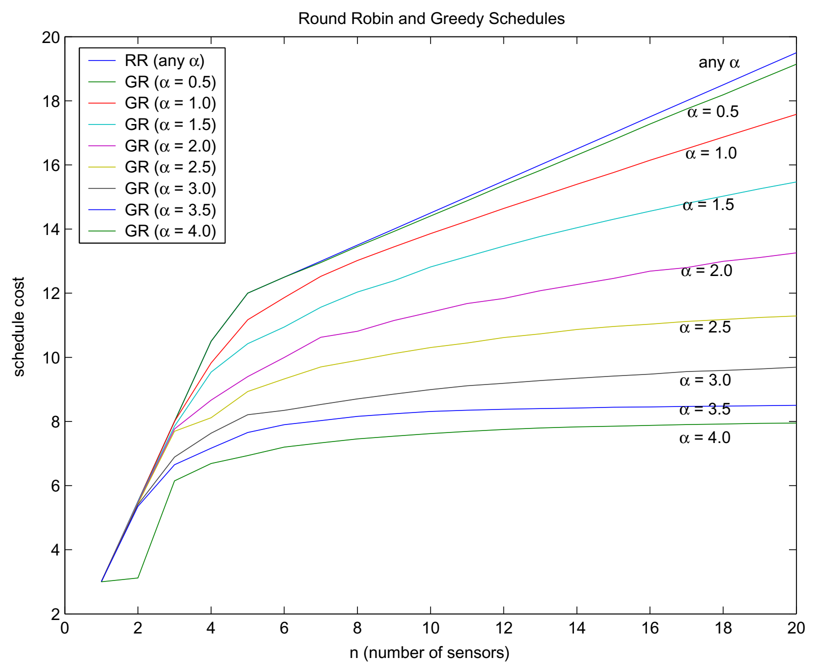


Figure 2.1: Round-Robin and Original Greedy (Strategy 1 only); $\lambda_i = 2, \sigma_i = 7$, for all i.

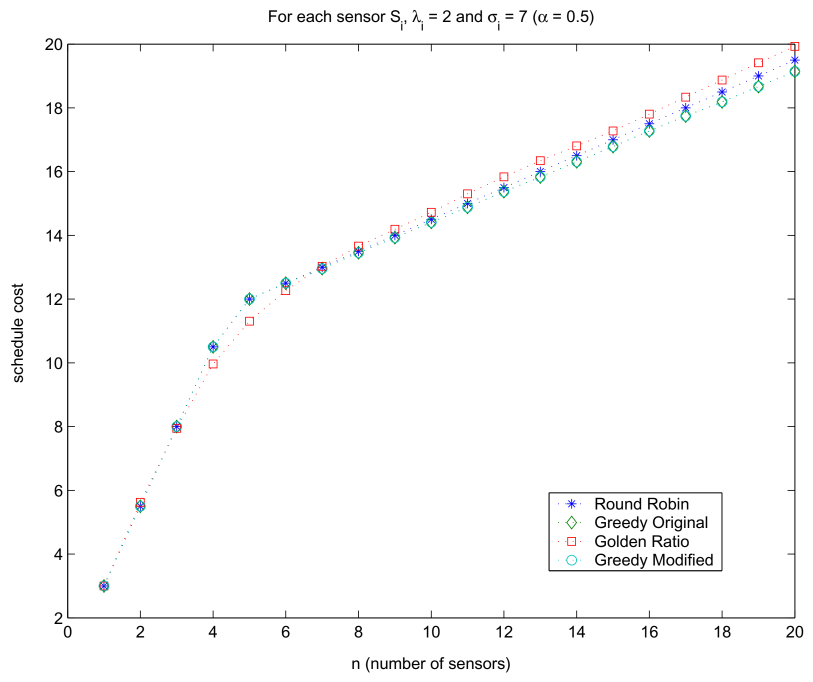


Figure 2.2: Given α and Fixed λ.

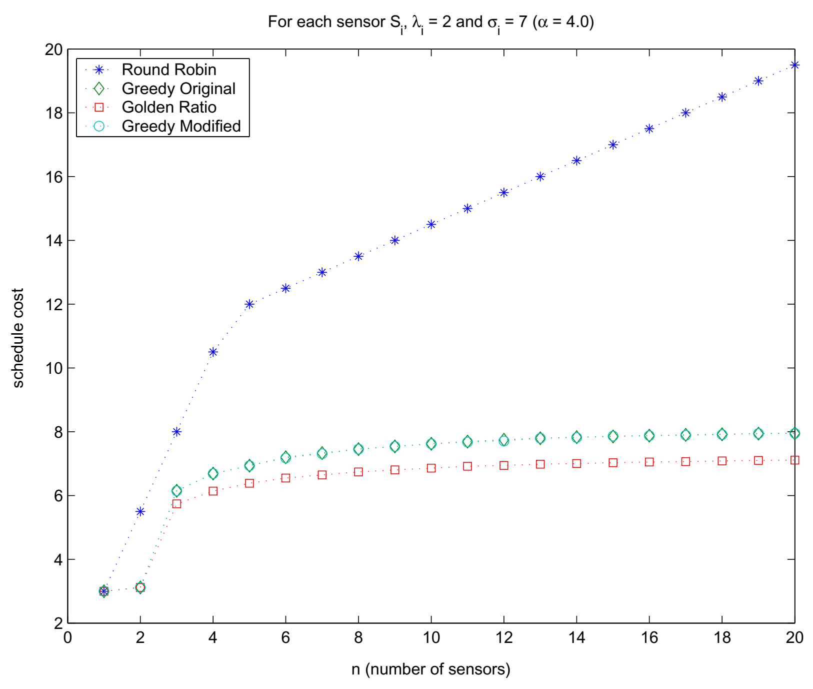


Figure 2.3: New α and Fixed λ.

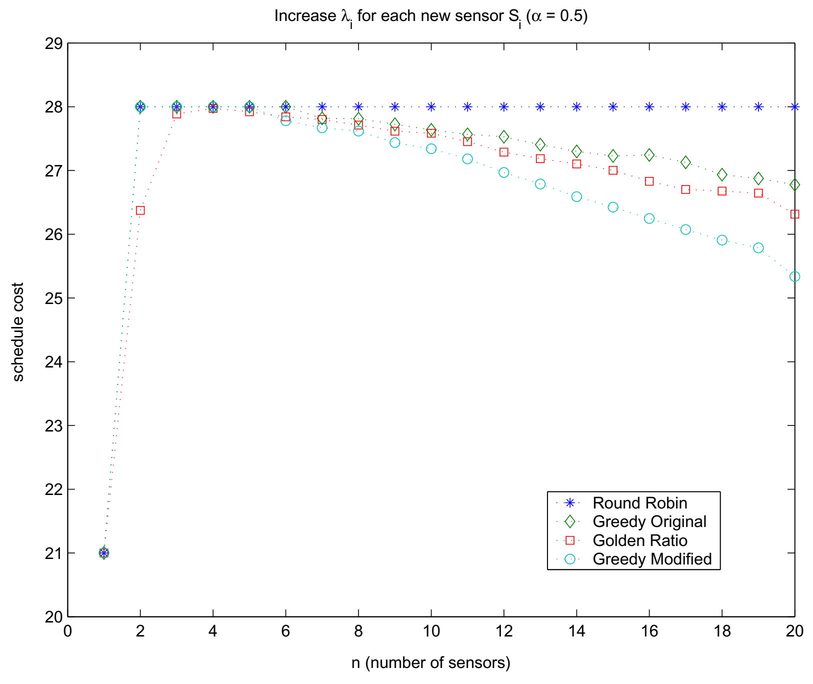


Figure 2.4: Given α; λ_i Increasing with i; $\sigma_i = 7$ for all i.

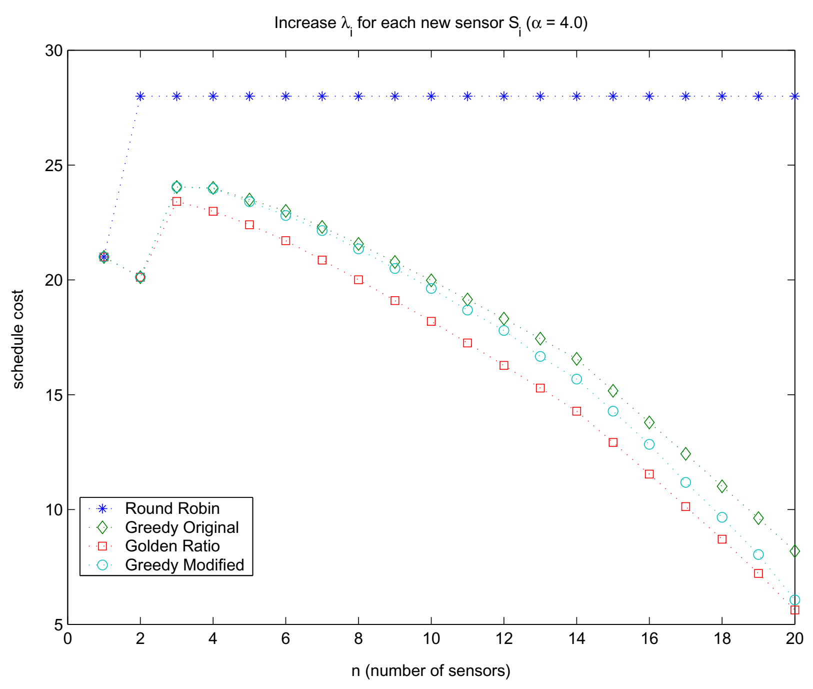


Figure 2.5: New α; λ_i Increasing with i; $\sigma_i = 7$ for all i.

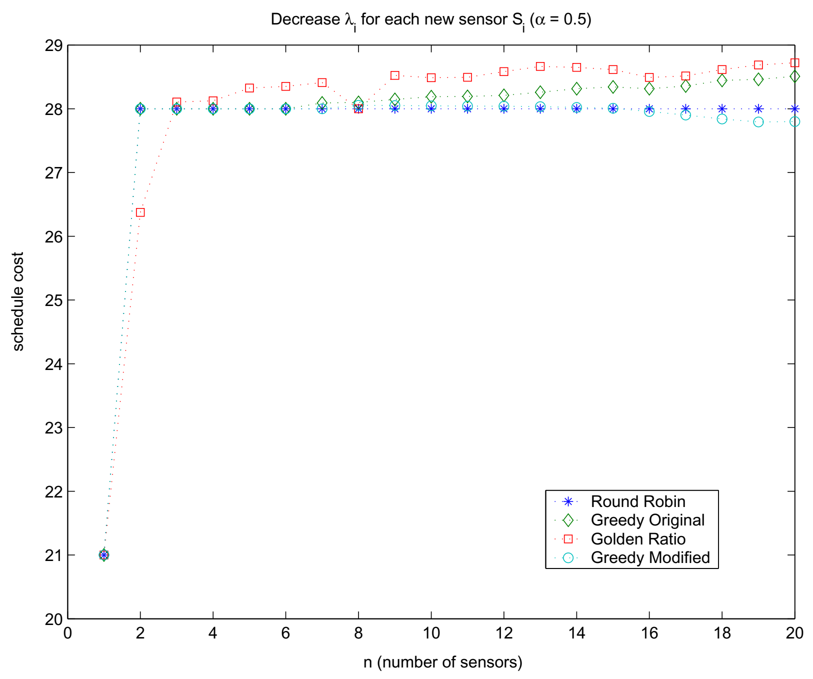


Figure 2.6: Given α; λ_i Decreasing with i; $\sigma_i = 7$ for all i.

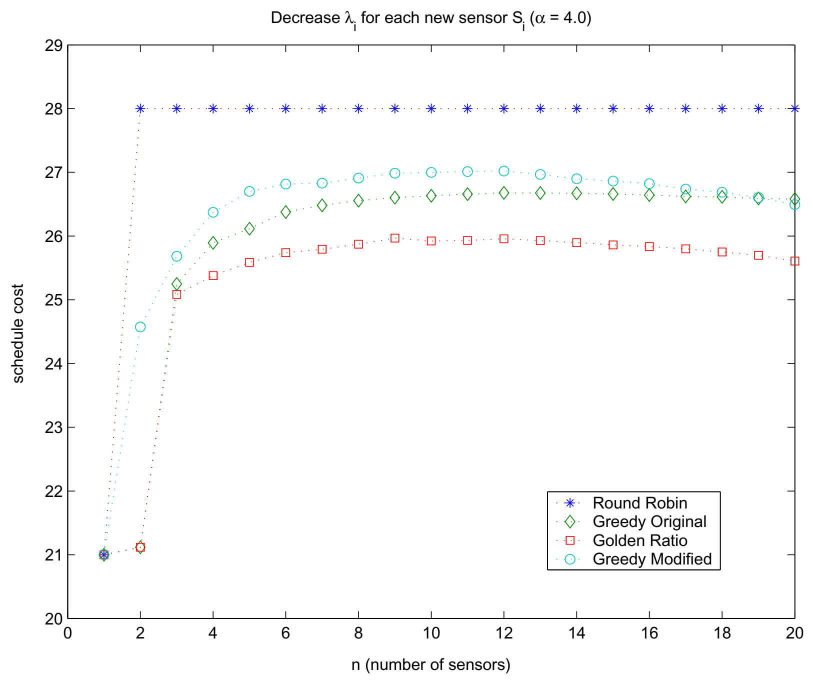


Figure 2.7: New α; λ_i Decreasing with i; $\sigma_i = 7$ for all i.

2.6.3 Various Experiments

Figure 2.1 compares Round-Robin schedule with Original Greedy Rule schedule, based on *strategy 1*, for various values of α. The listening cost and the reboot cost for every sensor are fixed, and equal to 2 and 7 respectively. As is evident from the plot, Round-Robin schedule completely ignores p_i and schedules switches according to μ_i. For large α, Greedy Rule schedule outperforms Round-Robin. This implies that the greedy algorithms are more effective when the popularity of missions varies widely.

Figure 2.2 and figure 2.3 address cases when $\alpha = 0.5$ and $\alpha = 4.0$, respectively. Four schedules are considered; σ_i and λ_i are independent from i. Observe that for larger α the greedy rules outperform simple Round-Robin, as expected. Furthermore, the greedy rules behave identically, whether they use *strategy 1* or *strategy 2* to compute the new q_i.

This happens because the new q_i would be the same in both strategies.

Figure 2.4 and figure 2.5 are based on the assumption that $\alpha = 0.5$ and $\alpha = 4.0$ respectively, λ_i increases with i, and $\sigma_i = 7.0$. Four schedules are considered. Evidently, the greedy rules outperform simple Round-Robin, regardless of the α. Furthermore, the greedy rule with *strategy 2* is better, since it takes into account listening cost to modify the q_i.

Figure 2.7 is based on the assumption that λ_i decreases when i increases, $\alpha = 4.0$, and $\sigma_i = 7$. Four schedules are considered. For this α, greedy rules outperform Round-Robin. We have performed the same experiment with $\alpha = 0.5$, observing only minor variations in performance across this four-schedule set. The results are depicted on figure 2.6.

2.6.4 Insights from Simulation

The Round-Robin schedule completely ignores p_i, and schedules switches according to μ_i; the Round-Robin cost is independent of α. When all λ_i are equal, the Original and Modified Greedy rules are insensitive to the choice of strategy (1 or 2), but the Golden-Ratio Rule is the best. As expected, for α sufficiently large, the greedy rules outperform Round-Robin, as they take into account the listening cost λ_i while calculating q_i.

2.7 Conclusion and Future Work

Our admission of the sensor rebooting cost into the model leads to several questions like when a sensor needs to be placed in a sleep mode or when should it stay on even when there is no data available for that sensor on a broadcast channel.

We have derived a lower bound for the minimum cost of the schedule. It remains an open problem, though, to employ this lower bound to approximate the minimum cost within a ratio that could be quantified analytically or numerically. Designing a greedy solution that is a 2-approximation to the fractional solution would be interesting.

Our assumptions are that all info-pages are unit-sized, and that they are disseminated over a single broadcast channel. It would be interesting—theoretically as well as in terms of realistic applications—to relax these assumptions so as to allow longer info-pages and multiple broadcast channels.

Modeling page service priorities and dependencies among pages as well as admitting non-linear obsoleteness cost functions would make the model more realistic.

Chapter 3

You Can't Get There From Here: Sensor Scheduling with Refocusing Delays

This chapter[1] presents a problem of sensor scheduling in sensor networks employed in monitoring and surveillance applications. The goal is to schedule sensors to observe sites so as to reduce potential loss of limited observation due to time delay necessary to pan and tilt the sensors.

3.1 Introduction and Motivation

Area monitoring is a common application of wireless sensor networks. In wireless sensor networks employed in monitoring or surveillance applications, individual sensors may perform pre-assigned or on-demand tasks. In particular, sensors such as visual cameras,

[1]The work of this chapter is published in [6].

radars, or passive infrared cameras may need to observe distinct geographical locations (or sites). It is usually the case that the number of tasks (or observations) to perform is larger than the number of sensors in the network. Hence, sensors' time must be shared to observe multiple sites. In addition, some observations may be more important than others. Thus, a schedule is required in order to specify which sensor observes what site at each particular time. This chapter studies a problem of scheduling a single sensor to observe n distinct sites. Our work is motivated by the research of Yavuz and Jeffcoat [62], [63].

More formally, in the sensor scheduling problem we have m sensors (cameras, radars, PIRs, etc.) that need to observe $n > m$ distinct locations. The objective is to schedule the sensors to observe the most important sites as frequently as possible, in order to minimize the amount, or value, of information we fail obtain when particular sites are *not* observed. Applications of this kind of problem are common. For instance, camera surveillance may be used to monitor intrusions along a border, in which case there may be many distinct unprotected places along the border to observe, to protect against use by intruders. If it is too costly to dedicate a single camera to each site, then one or more cameras may be scheduled to alternate between observation of different locations.

Consider an example in which a sensor is required to observe sites A, B, and C. The sensor can only focus on one site at a time, say site A, collecting all the new potential information from that site. The other two sites B and C are left unobserved, in which case we must rely on earlier observations of them. Each time a site goes unobserved, we lose some new information. Two types of costs can be associated (within a given timeslot) with this lost information: 1) a *fixed cost* of not observing a particular site at a particular time, and 2) a *variable cost* that is a linear function of the time gap since its last visitation. Both costs provide an incentive to visit the site. The motivation of the second type of cost is to model the fact that information becomes increasingly obsolete over time. The goal is to construct a periodic schedule (i.e. a schedule that repeats itself every $T \in \mathbf{Z}^+$ timeslots), that minimizes the average cost of lost information per timeslot.

The crucial aspect of this chapters's problem formulation is that sometimes a sensor may not be able to observe two distinct sites right after each other as it needs time to adjust and refocus. Prior research has neglected such time delays. Though time delay to refocus the sensor to a new location has a considerable impact on a feasible optimal schedule. Refocusing time delay may preclude many sites from being scheduled next. Suppose that in the example above it takes one idle timeslot to refocus the sensor from one site to another. Say a sensor is observing site A during the last timeslot. What should a sensor do during the next timeslot? Observing site B or C is not possible since it takes time for a sensor to refocus. The sensor may either continue observing A in the next timeslot or not observe anything in the next timeslot, in which case it takes one timeslot to refocus to another site in order to observe it in the second timeslot after the idle timeslot.

Throughout the rest of the chapter we will refer to a sensor as a camera since camera is a typical sensor used for observations. Conserving the energy required for movement could be a motivation as well, although we defer optimizing for energy to future work. The rest of this chapter is organized as follows. Section 3.2 discusses some related work. In Section 3.3, we formulate different variants of a single sensor scheduling problem with delay constraints and prove some hardness and structural properties. In Section 3.4, we derive a lower bound on the optimal solution cost for the sensor scheduling problem. We derive exact solution to some special case settings in Section 3.5. Then in Section 3.6 and Section 3.7 we propose greedy-based algorithms and evaluate them on synthetic data. We conclude by discussing future work in Section 3.8.

3.2 Related Work

A min-max variant of the Single-Sensor Scheduling Problem (SSSP) is studied by practitioners in [63], [62]. In [62], a formulation minimizing the maximum information loss for any site and in any timeslot is studied; it is NP-hard, and heuristic algorithms are given. In that formulation, it is assumed that time transition between sites is negligible. Our relaxation of this assumption is a crucial aspect of the problem we study in the present work. Because of the delays between observations of various site pairs, only a subset of sites in general are reachable in the next timeslot at any given time, which separates our problem from the previously studied Broadcast Disks and Maintenance problems. Other works on either single sensor scheduling or multi-sensor scheduling can be found in [36], [37], [57], [22]

The min-sum sensor scheduling problem that we define here is a generalization of the Broadcast Disks [4] problem. In that problem, there is a quadratic cost paid (in total) for time gaps between receiving pages, and the objective is to minimize the sum of costs. In our problem, there are quadratic costs for gaps (between observations of sites) as well as *linear* costs for gaps. (Equivalently, during each timeslot within a site's gap, there is one penalty linear in the time since the gap's last observation, as well as one penalty that is a simple constant.) The Broadcast Disks problem does not consider time delay constraints that may preclude many pages from being scheduled next.

An equivalent problem of Broadcast Disks is the problem of scheduling for Teletext systems [11, 12]. The single-sensor (SSSP) special case of the sensor scheduling problem that we focus on in this work, in which there is only one sensor scheduled to observe n sites, generalizes the Teletext problem, in the same two ways as above, viz., by adding fixed costs and delay constraints. In [12], Ammar and Wong show that there always exist optimal

Teletext schedule solutions that are periodic; in [11], they show how to construct broadcast cycles. They also derive a square root rule according to which an item i's broadcasting frequency is proportional to the square root of i's request probability, which in our case corresponds roughly to normalized variable costs.

Also closely related is the Machine Maintenance problem [17]. In that problem, which is another generalization of Broadcast Disks, a (linear) operating cost is charged based on the time elapsed since the last service, while a constant maintenance cost is charged for each timeslot t when the machine *is* serviced. Note that the maintenance cost is charged in exactly the complement of the timeslots when our fixed cost is, whereas the operating cost corresponds to our variable cost. (There are no switching delays in the Maintenance problem.) Bar-Noy et al. [17] prove that the maintenance scheduling problem is NP-hard and provide approximation algorithms. Note that hardness for the single-sensor version of our problem does not immediately follow from this result. We give an unrelated hardness proof in the chapter.

More broadly, various other scheduling problems involve the notion of travel delays between sites. In hard disk scheduling [54], e.g., the Shortest Seek Time First algorithm chooses the request closest to the current head position, in order to minimizes latency. In the k-server problem [29], k servers will service client requests, as they appear online, within a metric space. The goal is to minimize the total distance moved by the servers servicing requests. In the related offline Watchman problem [44], one server must choose a short route between a set of locations to guard, avoiding obstacles in the region. The SSSP differs from these problems, however, in that the switching delays are hard constraints rather than soft. That is, these delays rule out certain sequences of site observations as infeasible. We seek low-cost periodic schedules observing these restrictions.

Another related work [28] studies a problem where a single camera is required to observe multiple people. It does so by dividing the camera's time in order to view everyone. In their work an additional tracking camera with a fixed wide field of view is employed to detect, track, and classify moving targets. The information of the targets is then provided to the active camera that can pan, tilt, and zoom to collect high resolution video. The scheduling problem consists of deciding which person the active camera will focus its sensing resources on until the next state update. The arrival times of people entering the scene is not known in advance making it an online scheduling problem.

The work in [60] is an extension of [28] to multiple active cameras scheduling. This work studies a system of multiple pan-tilt-zoom (PTZ) cameras, assisted by a fixed wide-angle camera, to collect high resolution video of the (many) moving targets. The system first assigns a subset of the requests or targets to each camera. The cameras then select the parameter settings that best satisfy the assigned competing requests to provide high resolution views of the moving objects. The main difference in our work is that the targets are not moving objects but stationary geographical locations with associated costs due to lost information.

3.3 Model

In the Single-Sensor Scheduling Problem (SSSP), we have one sensor that observes a collection of n sites at discrete time intervals. In each timeslot, the sensor can choose (at most) one site to observe. The problem is to find a periodic schedule (with some period T), minimizing total costs, as described below. Initially we will assume that the time for switching from one site to another is negligible. Later we will incorporate this delay into the model.

3.3.1 Preliminaries

Our model uses the following notation:

- a_i – fixed penalty for not observing site i
- $b_{i,t}$ – variable penalty for not observing site i at time t
- $y_{i,t}$ – time of last observing site i before time moment t, set to t iff the sensor is observing site i at time slot t, and otherwise equals $y_{i,t-1}$
- $g_{i,t} = (t - y_{i,t})$ – the time length (or gap) since last observation of site i, prior to time t

Let $x_{i,t}$ be a decision variable taking value 1 if the ith site is observed at time t ($1 \leq t \leq T$), and 0 otherwise. The penalty for not observing a site i at time t is expressed as follows:

$$a_i(1 - x_{i,t}) + b_{i,t}(t - y_{i,t})$$

A variant of the sensor scheduling problem was formulated for one-sensor case in [63, 62]. In their formulation, the objective was to minimize the maximum cost $a_i(1 - x_{i,t}) + b_{i,t}(t - y_{i,t})$, among all sites i *and* times t, for a schedule defined over period T. The factor $(t - y_{i,t})$ is the gap $g_{i,t}$, the length of time since the last observation of site i. Following the Broadcast Disks and Maintenance problems, however, we optimize for the average (or equivalently, the total) penalty per slot, over all sites i and times t. The parameter $b_{i,t}$ could be tuned based on the needs of the application, e.g. increased during rush hours or decreased otherwise if the activity level of site i at time t diminishes. For the bulk of the chapter, however, we will assume for simplicity that $b_{i,t}$ as a time-invariant parameter b_i.

Thus, our problem is specified by the integer program formulation given in Table 3.1.

$$\min \frac{1}{T}\sum_{i=1}^{n}\sum_{t=1}^{T} a_i(1-x_{i,t}) + b_i(t-y_{i,t})$$

$$\text{s.t.} \sum_{i=1}^{n} x_{i,t} \leq 1 \qquad \forall_{t\in\{1,\dots,T\}} \tag{3.3.1}$$

$$0 \leq y_{i,t} - y_{i,t-1} \leq t \cdot x_{i,t} \qquad \forall_{i\in\{1,\dots,n\}} \; \forall_{t\in\{1,\dots,T\}} \tag{3.3.2}$$

$$tx_{i,t} \leq y_{i,t} \leq t \qquad \forall_{i\in\{1,\dots,n\}} \; \forall_{t\in\{1,\dots,T\}} \tag{3.3.3}$$

$$x_{i,0} = 1 \qquad \forall_{i\in\{1,\dots,n\}} \tag{3.3.4}$$

$$y_{i,0} = 0 \qquad \forall_{i\in\{1,\dots,n\}} \tag{3.3.5}$$

$$x_{i,t} \in \{0,1\} \qquad \forall_{i\in\{1,\dots,n\}} \; \forall_{t\in\{1,\dots,T\}} \tag{3.3.6}$$

$$d_{i,j}x_{i,t} \leq \tau + (1-x_{j,t-1})\Delta \qquad \forall_{i\in\{1,\dots,n\}} \; \forall_{j\in\{1,\dots,n\}} \; \forall_{t\in\{1,\dots,T\}} \tag{3.3.7}$$

Table 3.1: IP formulation for SSSP.

The first constraint prevents multiple sites from being observed at the same time. The second and third constraints of the formulation above ensure that $y_{i,t}$ takes on value t if an observation of site i occurs at time t, and otherwise equals $y_{i,t-1}$. The fourth and fifth constraints dictate that all sites are treated as having been observed at time 0. In some of our experiments, we relax these constraints and allow $x_{i,0}$ and $y_{i,0}$ to be specified in the input, which allows us to code for certain sites having initially already been unobserved for some larger number of timeslots.

The sixth constraint restrict the $x_{i,t}$ variables to 0 or 1 values. We do not restrict the values of the $y_{i,t}$ variables, allowing them to take on negative values when the $y_{i,0}$ values are specified as described above. The $y_{i,t}$ will, however, always take on integer values due to the second and third constraints, and so a constraint of the form $\mathbf{y_{i,t}} \in \mathbf{Z}$ is superfluous. The seventh constrain is a delay constraint which is explained in the next subsection.

3.3.2 Delay Constraints

Let $\mathcal{D}$ be an $n \times n$ matrix with entries $d_{i,j}$ corresponding to transition times (in units of timeslots) between sites j and i. ($d_{i,j} = 0$ indicates that the sensor can observe site i after observing site j, without requiring any idle slots.) We can then formulate the constraint as:

$$d_{i,j} x_{i,t} \leq (t - y_{j,t-1} - 1) \tag{3.3.8}$$

Recall that $(t-1) - y_{j,t-1}$ is the value of gap $g_{i,t-1}$, which is the length of time since the last observation of site j prior to time $t-1$. The right hand side is nonnegative since the gap cannot be negative. If i is not scheduled at time t, then the constraint is satisfied trivially because the LHS is 0; if it is, then the LHS is the transition time from the previously scheduled site j to the current site i. Note that for the previously scheduled j, the RHS is smaller than for the j that was not scheduled previously. Consider the following example, in which we have sites A, B, C, and D, and transition matrix $\mathcal{D}$ as shown in Table 3.2.

	A	B	C	D
A	0	0	1	2
B	0	0	0	1
C	1	0	0	0
D	2	1	0	0

Table 3.2: Transition matrix $\mathcal{D}$.

Assume the schedule up to now has been $A\ B\ C\ A$. Now we need to make a decision for the next timeslot, $t = 5$. Note that the previously scheduled site j at time $t-1$ is A, and that $g_{j,t-1} = (t-1) - y_{j,t-1} = 4 - y_{A,4} = 0$. For any other j the gap will be even greater, and so therefore will the right hand side. Thus the RHS of the constraint equation 3.3.8 is 0. This means that on the LHS we must have 0 to satisfy the constraint. Hence, only those sites in row A with 0 entries can be considered (i.e., A and B) for timeslot $t = 5$. Now, suppose we leave $t = 5$ idle. This would give us schedule $A\ B\ C\ A\ \Box$ (where $\Box$ indicates no site observed), with our next scheduling decision for timeslot $t = 6$. Note that the (unique) minimum $g_{j,t-1} = (t-1) - y_{j,t-1}$ is $5 - y_{A,5} = 1$. Therefore the RHS is 1. The LHS should have a site scheduled at timeslot $t = 6$ that is reachable in time at most 1 (i.e., A, B, or C). Similarly, if we leave timeslot $t = 6$ idle with the current schedule so far as $A\ B\ C\ A\ \Box\ \Box$, then for $t = 7$ the (unique) minimum $g_{j,t-1}$ is due to A and is equal to 2. Hence, at timeslot $t = 7$ we can schedule any site reachable from A within 2 timeslots (i.e., A, B, C, or D).

In most of this chapter we restrict our attention to a special case in which the entries $d_{i,j}$ are 0 or 1, meaning the sites are reachable within at most one idle slot. The constraint can now be formulated as:

$$d_{i,j} x_{i,t} \leq 1 - x_{j,t-1}$$

If, however, we let the $d_{i,j}$ be the time delay parameters themselves (rather than units of timeslots) associated with the refocusing of the sensor from site j to site i, set $\Delta = \max_{i,j} d_{i,j}$, and furthermore let τ be the time limit during which the sensor should finish its refocus. Then we can reformulate the constraint to incorporate the pan and tilt delay as: $d_{i,j}x_{i,t} \leq \tau + (1 - x_{j,t-1})\Delta$. Both constraints are equivalent: one expresses the delay in terms of number of slots, and the other expresses it in terms of time.

3.3.3 Hardness and Periodicity

We prove hardness by means of a Cook reduction from the classical Maximum Independent Set (MIS) problem, which is NP-hard [31].

Theorem 3.3.1. *Solving SSSP optimally is NP-hard.*

Proof. Given an instance $G = (V, E)$ of MIS, we create a family of SSSP instances, indexed by value T ranging from 1 to $n = |V|$, as follows. For each node $v_i \in V$, we introduce a site i, with constant cost $a_i = 0$ and linear cost $b_i = 1$. For each edge $(v_i, v_j) \in E$, we set the delay between nodes i and j to be infinity (or some sufficiently large constant); for each edge not present, we set the corresponding delay to zero. Assume we have solved all n SSSP instances optimally.

We distinguish between three possible situations that a given site may be in, within a particular problem solution: it may be observed zero times, one time, or multiple times. The difference in cost between zero observations and one observation is much greater than the difference between one observation and multiple observations: zero observations will incur a cost quadratic in both T and the number of times the schedule cycles; one observation will incur a cost at most quadratic in T and linear in the number of cycles. The bulk of the solution cost will depend on the number of sites not observed at all.

We note that for any value T over the specified range, there will exist a feasible solution visiting a site in every timeslot: assuming the delay between a site and itself is zero, a schedule observing the same site in every timeslot will always be feasible. By the argument above, however, sites will only be observed multiple times when it is impossible to add some other zero-visited site to the schedule. For a sufficiently small value of T, therefore, we will obtain a schedule in which all sites observed are observed only once. It is clear from the construction that the sites observed in the SSSP optimal solution of the instance with the largest such value T will form a maximum independent set in the underlying graph. $\square$

We also note that we can restrict our attention to periodic schedules in the following sense, assuming that delay constraints are symmetric, i.e., the delay for moving from site i to j is the same as for moving from j to i.

Theorem 3.3.2. *Every SSSP problem instance with symmetric delay constraints admits an optimal solution that is periodic.*

Proof. (sketch) As with the case of the Maintenance Problem [17], a proof can be given by adapting the argument of Anily et al. [13]. The proof begins by bounding from above the distance between two consecutive observations of any given site in an optimal solution; indeed, if there is a longer interval between the two observations, then the schedule could be improved by inserting a third, intervening observation of the site. Since there are then only a finite number of possible *states*, any schedule can be transformed into a periodic schedule at least as good. $\square$

3.4 Lower Bound

In this section we derive a lower bound on the optimal solution cost for the single sensor scheduling problem. Assume that the schedule is perfectly periodic, that is, each site i is visited periodically with a fixed period τ_i. (We will show later that this assumption is justified.) We ignore for now the transition delay matrix $\mathcal{D}$ whose entries $d_{i,j}$ dictate schedulability of the sites, given the previously scheduled site. As in [17], we give a nonlinear relaxation to the sensors scheduling problem. Since the introduction of delay constraints only increases the optimal solution cost, the lower bound still holds (albeit less tightly) when delays are present.

Proposition 3.4.1. *Suppose the site i is scheduled at timeslot t and is not scheduled in timeslots $t+1,\dots,t+x-1$. Then the total variable cost incurred by scheduling site i for the x timeslots $t,\dots,t+x-1$ is given by $(b_i/2)(x-1)x$.*

Proof. Site i incurs a variable cost of $b_i \cdot j$ in timeslots $t+j$, for $j=0,\dots,x-1$, so the total variable cost incurred by site i in timeslots $t,\dots,t+x-1$ is given by:

$$\sum_{j=0}^{x-1} b_i \cdot j = \frac{b_i}{2}(x-1)(x)$$

□

Proposition 3.4.2. *Suppose the site i is scheduled in timeslot t and is not scheduled in timeslots $t+1,\dots,t+x-1$. Then the total fixed cost incurred by scheduling site i for the x timeslots $t,\dots,t+x-1$ is given by $(x-1)a_i$.*

Proof. Site i incurs a fixed cost of a_i in timeslots $t+j$, for $j=1,\dots,x-1$, and 0 cost in timeslot t, so the total fixed cost incurred by site i in timeslots $t,\dots,t+x-1$ is:

$$0+\sum_{j=1}^{x-1} a_i = (x-1)(a_i)$$

□

The next proposition shows that the lower bound schedule is indeed a perfectly periodic schedule.

Proposition 3.4.3. *Lower bound schedule is* perfectly *periodic.*

Proof. Given is a schedule with time horizon T. Suppose site i is scheduled at time t and not scheduled at times $t+1, \ldots, t+x-1$, then again is scheduled at time $t+x$, and not scheduled at times $t+x+1, \ldots, t+x+y-1$. In other words, the schedule consists of site i, followed by a skip of $x-1$, followed by site i, followed by a skip of $y-1$, so that there are two periods: x and y. We will show that on the time horizon T, either $x = y$ or a better schedule exist where the period $z = \frac{x+y}{2}$. If $x = y$, then the costs of the first x slots and of the second y slots are the same, and thus is perfectly periodic, so assume otherwise. That is, when $x \neq y$, we have $x^2 + y^2 - 2xy = (x-y)(x-y) > 0$. Multiplying both sides by b and grouping terms, we get

$$(2bx^2 - bx^2) + (2by^2 - by^2) + (2bx - 2bx) + (2by - 2by) - 2bxy > 0$$

Dividing both sides by $4(x+y)$ yields

$$\frac{bx^2 - bx + by^2 - by}{2(x+y)} - \frac{bx}{4} - \frac{by}{4} + \frac{b}{2} > 0$$

Rearranging the terms, we get

$$\frac{b(x^2 - x + y^2 - y)}{2(x+y)} > \frac{bx + by}{4} - \frac{b}{2}$$

Similarly,

$$\frac{b(\frac{x(x-1)}{2} + \frac{y(y-1)}{2})}{x+y} > \frac{bx + by}{4} - \frac{b}{2}$$

The LHS is the cost associated with b for non-regular schedule. The RHS is the cost associated with b for a regular schedule with $z = \frac{(x+y)}{2}$. The costs associated with a for both schedules on the time horizon T are the same. Thus the regular schedule indeed has lower cost, which completes the proof.

□

Now consider the following nonlinear program with variables $\tau_1, \dots, \tau_n$.

$$\begin{array}{ll} \text{min:} & \sum_{i=1}^{n} \frac{b_i(\tau_i - 1)}{2} + \sum_{i=1}^{n} (a_i - \frac{a_i}{\tau_i}) \\ \text{s.t.:} & \sum_{i=1}^{n} \frac{1}{\tau_i} \leq 1 \\ & \tau_i \geq 1 \quad \forall\, i \end{array} \tag{3.4.1}$$

Note that the average cost of the schedule as time goes to infinity is equivalent to average cost over the period, and thus the total cost over the period $\frac{b_i(\tau_i-1)\tau_i}{2} + a_i(\tau_i - 1)$ is being divided by the period τ_i. In the nonlinear program we schedule each site in fixed periods such that the average cost per slot is minimized. This is a relaxation because these periods may not be integers. Furthermore, even if we round these periods to integers, the schedule may not be achievable simultaneously for all the sites, since more the one site will need to be scheduled in same timeslot for some timeslots. Since the a_i terms in the second summation are constants, we can change the objective function to:

$$\begin{array}{ll} \text{min:} & \sum_{i=1}^{n} \frac{b_i(\tau_i - 1)}{2} - \sum_{i=1}^{n} \frac{a_i}{\tau_i} \\ \text{s.t.:} & \sum_{i=1}^{n} \frac{1}{\tau_i} \leq 1 \\ & \tau_i \geq 1 \quad \forall\, i \end{array} \tag{3.4.2}$$

The objective function is concave with convex constraints. We can use Lagrangian relaxation to solve for optimal $\tau_1, \dots, \tau_n$.

Theorem 3.4.4. *An optimal solution to the nonlinear relaxation is given by* $\tau_i = \sqrt{\frac{2(\lambda^* - a_i)}{b_i}}$*, where* $\lambda^* > \max_i(a_i)$ *and* $\lambda^* > \frac{b_i + 2a_i}{2}$ $(1 \leq i \leq n)$*.*

Proof. We obtain the following nonlinear program by applying Lagrangian relaxation to Equation 3.4.2.

$$\text{min: } \sum_{i=1}^{n} \frac{b_i(\tau_i - 1)}{2} - \sum_{i=1}^{n} \frac{a_i}{\tau_i} - \lambda\Big(1 - \sum_{i=1}^{n} \frac{1}{\tau_i}\Big) - \sum_{i=1}^{n} \mu_i(\tau_i - 1) \qquad (3.4.3)$$

For any fixed $\lambda \geq 0$ and $\mu_i \geq 0$ for $1 \leq i \leq n$, the optimal value of the Lagrangian relaxation 3.4.3 is a lower bound on the optimal value of the nonlinear program 3.4.2, and the optimal solution of relaxation 3.4.3 is given by $\tau_i = \sqrt{\frac{2(\lambda - a_i)}{b_i - 2\mu_i}}$ $(1 \leq i \leq n)$, provided that $b_i - 2\mu_i > 0$ (by taking partial derivatives of the Lagrangian with respect to τ_i).

In order to find global minima, we need to satisfy Karush-Kuhn-Tucker conditions. The constraints could either be loose or tight. If the constraints are tight then the corresponding Lagrange multipliers will be positive, whereas if the constraints are loose then the corresponding Lagrange multipliers will be 0. None of the constraints of $\tau_i \geq 1 \ (1 \leq i \leq n)$ can be tight, since if at least one $\tau_i = 1$ then the constraint of $\sum_{i=1}^{n} \frac{1}{\tau_i} \leq 1$ will only be satisfied when all the other $\tau_j = \infty \ (j \neq i)$. Thus, all the constraints $\tau_i \geq 1 \ (1 \leq i \leq n)$ are loose (i.e., $\tau_i > 1 \ (1 \leq i \leq n)$) and, hence, all $\mu_i = 0 \ (1 \leq i \leq n)$. On the other hand the constraint $\sum_{i=1}^{n} \frac{1}{\tau_i} \leq 1$ is tight, and hence $\lambda > 0$. Let λ^* be the value of λ. In fact, since $\mu_i = 0$, in order for $\tau_i = \sqrt{\frac{2(\lambda - a_i)}{b_i - 2\mu_i}}$ $(1 \leq i \leq n)$ to have a real solution, λ^* must be at least $\max_i \{a_i\}$. In addition, in order to satisfy the loose constraints of $\tau > 1$ we need $\sqrt{\frac{2(\lambda - a_i)}{b_i}} > 1$, which is equivalent to $\lambda^* > \frac{b_i + 2a_i}{2}$ $(1 \leq i \leq n)$. □

In order to find such λ^* we can solve an equation $\sum_{i=1}^{n} \sqrt{\frac{b_i}{2(\lambda - a_i)}} = 1$. This equation can be solved using some variation of Newton's method. Let $q_i = \frac{1}{\tau_i}$. Observe that if we let all a_i be the same and equal to say a, then $\lambda^* = \frac{\left(\sum_{i=1}^{n} \sqrt{b_i}\right)^2 + 2a}{2}$. Furthermore, $q_i = \frac{1}{\tau_i} = \frac{\sqrt{b_i}}{\sum_{i=1}^{n} \sqrt{b_i}}$. We see that in the optimal solution with all a_i equal, $\frac{q_i}{q_j} = \frac{\sqrt{b_i}}{\sqrt{b_j}}$, as expected.

3.5 Special Cases

In general, the problem is NP-hard, so the best we can hope for is approximation algorithms. The problem can be solved optimally, though, for some special cases. The case of one sensor and one site is trivial since the optimal schedule is to observe the site perpetually, in which case there is trivially zero information loss. However, the case of a sensor scheduled to observe two distinct sites is interesting. Should the sensor switch between the site observations or not? If the sensor is required to switch between the site observations, then in what pattern must it do so in order to minimize the cost?

Let us first consider a case where variable costs are zero.

Proposition 3.5.1. *Suppose we have two sites S_1 and S_2 with refocus delays $d_{1,2}$ and $d_{2,1}$, no variable costs (i.e. $b_1 = b_2 = 0$), and fixed costs a_1 and a_2, with, say, $a_1 \geq a_2$. Then an optimal schedule is to perpetually observe site S_1.*

Proof. Observing site S_1 in a timeslot will incur cost a_2 per slot. Observing site S_2 in a timeslot will incur cost a_1 per slot. Not observing anything in a timeslot will incur cost of $a_1 + a_2$ per slot. Since $a_2 \leq a_1 \leq (a_1 + a_2)$, the minimum cost is obtained by observing S_1 in every timeslot. □

The result is easily extendable to a case with n sites and no variable costs, in which case the optimal schedule is still to observe a site with the biggest fixed cost.

Let us now consider a case where variable costs are not zero.

Proposition 3.5.2. *Suppose we have two sites S_1 and S_2 with refocus delays $d_{1,2}$ and $d_{2,1}$, variable costs $b_1 \neq 0$ and $b_2 \neq 0$, respectively, and fixed costs a_1 and a_2 respectively. Then an optimal schedule is of the form:*

$$((S_1)^x(\square)^{d_{2,1}}(S_2)^y(\square)^{d_{1,2}})^*$$

That is to say that the schedule is periodic with period $x + d_{2,1} + y + d_{1,2}$, where x and y are positive integers. The schedule observes site S_1 for x timeslots, followed by $d_{2,1}$ idle timeslots, followed by observation of site S_2 for y timeslots, followed by $d_{1,2}$ idle timeslots.

Proof. Assume to contrary that the schedule is to only observe one site, say S_1. Then the cost of not observing S_2 grows quadratically while the size of schedule grows linearly. So the cost associated with not observing S_2 is quadratic function divided by linear function. Thus, as schedule size goes to infinity the schedule cost per slot goes to infinity. Therefore, S_2 must be observed at some point. Similarly, if we only observe S_2 and not S_1, the schedule cost per slot goes to infinity as well. Therefore, S_1 must be observed at some point as well. The only way to observe both site is to switch between them which require idle slots. Therefore, the schedule is of the form as described above, that is, observe S_1 for some x timeslots where x is a positive integer. Then the sensor transitions to site S_2, which requires $d_{2,1}$ idle timeslots in the schedule. Then the sensor observes site S_2 for some y timeslots and transitions back to S_1, which requires another $d_{1,2}$ idle timeslots in the schedule. $\square$

The values of x and y depend on parameters a_1, a_2, b_1, b_2, $d_{1,2}$, and $d_{2,1}$. In order to choose optimal values of x and y, we set up a multivariate function $\mathcal{C}(x, y)$ to minimize, which depends on x and y and treats the rest of the parameters as constants. Let $D = d_{1,2} + d_{2,1}$. Solve the following minimization problem to find x and y.

$$\begin{aligned} \min \quad & \mathcal{C}(x,y) = \frac{a_1\cdot(D+y)+a_2\cdot(D+x)}{D+x+y} + \\ & + \frac{\frac{b_1}{2}\cdot((D+y+1)\cdot(D+y))+\frac{b_2}{2}\cdot((D+x+1)\cdot(D+x))}{D+x+y} \\ \text{s.t.} \quad & x \in \mathbf{Z}^+, y \in \mathbf{Z}^+ \end{aligned} \tag{3.5.1}$$

Example

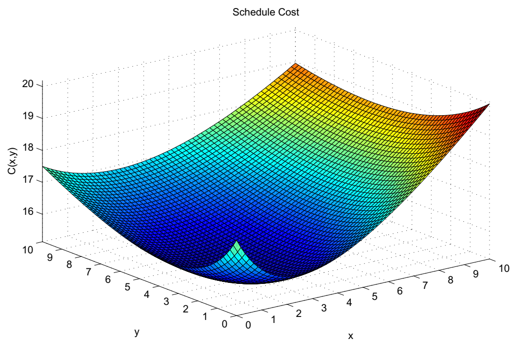


Figure 3.1: Schedule cost plot: two sites.

Problem: Given are two sites S_1, S_2 with the following costs: $a_1 = 8$, $a_2 = 2$, $b_1 = 1$, $b_2 = 2$. Let $d_{i,j} = d_{j,i} = 2$, so that transitioning between the sites takes two idle timeslots. Find x and y that minimizes the cost $\mathcal{C}(x, y)$.

Solution: We need to minimize the following function.

$$\mathcal{C}(x, y) = \frac{8(4 + y) + 2(4 + x)}{4 + x + y} + \frac{(1/2)(5 + y)(4 + y) + (5 + x)(4 + x))}{4 + x + y}$$

(See Figure 3.1.) This can be done using a tool Mathematica (i.e. $c = \mathcal{C}(x, y)$; $FindMinimum[\{c, x \geq 1 \,\&\&\, y \geq 1\}, \{x, y\}]$). The critical point is at $x_c = 2.16435$ and $y_c = 2.8287$ with $\mathcal{C}(x, y) = 15.3287$. Since x and y must be integers we test four possible cases: ($x = \lfloor x_c \rfloor$ or $x = \lceil x_c \rceil$ and ($y = \lfloor y_c \rfloor \, or (y = \lceil y_c \rceil$). The solution is $x = 2$ and $y = 3$ with $\mathcal{C}(x, y) = 15.3333$. Therefore, the schedule is $(S_1S_1\square\square S_2S_2S_2\square\square)^*$.

3.6 Algorithms

One heuristic approach is to relax the IP formulation, given in table 3.1, to obtain an LP-relaxation that could be efficiently solved, and then “round” the LP (fractional) solution to obtain a near-optimal solution. Unfortunately, solution to this LP-relaxation gives very loose bound. In fact, for increasing number of sites, the LP-relaxation optimizes the $y_{i,t}$ such that $(t - y_{i,t})$ for all i and t is 0 and the solver distributes fractional non-zero values among the $x_{i,t}$ such that the $\sum_{i=1}^{n} x_{i,t} \leq 1$. Fortunately, we have derived a tight lower bound in Section 3.4. We have implemented several greedy heuristics utilizing the solution of the LB given in Theorem 3.4.4.

The first greedy heuristic is called Greedy One-Step (see Algorithm 2). This heuristic tries to pick the next best site to visit based on the previously visited sites, as follows. It calculates a normalized frequency array q that dictates how often the sites must be scheduled based on a_i and b_i, where $q[i] = \frac{\sqrt{a_i+b_i}}{\sum_{i=1}^{n}\sqrt{a_i+b_i}}$. The idea behind this choice of $q[i]$ is due to the square-root rule, according to which if all a_i are the same then the $q[i]$ should be proportional to $\sqrt{b_i}$. However, if a_i are arbitrary, then it seems natural to try to incorporate the cost as the sum of a and b under the radical. Of course we can modify Algorithm 2 to calculate the frequency array by using τ_i's that are derived from the optimal solution of the nonlinear program 3.4.2 and let the $q[i] = \frac{1}{\tau_i}$. In fact, doing so may give better performance.

Notation 1. *In the following algorithms, let*
$\mathcal{S} = \{S_1, S_2, ..., S_n\}$ *be the set of all sites,* $\mathcal{D}$ *be the angle matrix with entries* $d_{i,j}$*,* τ *be the limit angle that a camera can sweep within 1 time slot,* $q[\cdot]$ *be the frequency array of schedulability,* $w[\cdot]$ *be the gap array, and* $r[\cdot]$ *be the potential cost array.*

The algorithm then proceeds as follows: at each step we keep track of the gap $w[i]$, which is the number of slots since the last appearance of the site i in the schedule. We schedule the site i which can be transitioned from previously scheduled site within one timeslot dictated by transition matrix $\mathcal{D}$ and that has the highest potential cost $r[i]$, which is calculated as $r[i] = (1 + w[i])q[i]$. We modified the algorithm to use $q[i] = \frac{1}{\tau_i}$, where τ_i

```
for each site $S_i \in \mathcal{S}$ do
  $w[i] \leftarrow 0, r[i] \leftarrow 0$
  $q[i] \leftarrow \frac{\sqrt{a_i+b_i}}{\sum_{i=1}^{n}\sqrt{a_i+b_i}}$ (or $q[i] \leftarrow \frac{1}{\tau_i}$ where $\tau_i$ is a solution to equation (3.4.2) of the lower bound)
end for
for $t = 1$ to $T$ do
  for each site $S_i \in \mathcal{S}$ do
    $r[i] \leftarrow (1 + w[i])q[i]$
  end for
  choose a site $m$ maximizing $r[m]$ and $d_{i,j}[m][j] \leq \tau$, where $j$ is the previously scheduled site
  schedule $S_m$
  for each site $S_i \in \mathcal{S}$ do
    $w[i] \leftarrow w[i] + 1$
  end for
  $w[m] \leftarrow 0$
  $t \leftarrow t + 1$
end for
```

Algorithm 2: Greedy One-Step Algorithm.

are derived from the solution to the lower bound given in Theorem 3.4.4.

The problem with the Greedy One-Step Algorithm is that if two important sites, say A and D, are far away from each other separated by unimportant sites, say B and C, then in order to transition from A to D, the sensor needs to visit B in the next slot then C in the second slot and only then D in the third slot. It may however be more optimal not to schedule anything during the next slot and visit D right away during the second slot since in our formulation the sensor can use the entire idle slot just to do the transition to any other site. We have implemented a greedy solution that does lookahead: *Greedy Two-Steps Lookahead.* The algorithm allows for idle slots. In essence it tries to consider the next two best sites (one after another) and compare it with scheduling an idle slot with the next best site. It schedules an idle or non-idle slot next depending on whichever gave better performance. The algorithm considers sites as follows. Let $w_2 \leftarrow w$. For the next slot

```
for each site $S_i \in \mathcal{S}$ do
    $w[i] \leftarrow 0, r[i] \leftarrow 0$
    $q[i] \leftarrow \frac{\sqrt{a_i+b_i}}{\sum_{i=1}^{n}\sqrt{a_i+b_i}}$ (or $q[i] \leftarrow \frac{1}{\tau_i}$ where $\tau_i$ is a solution to equation (3.4.2) of the lower
    bound.
end for
for $t = 1$ to $T$ do
    for each site $S_i \in \mathcal{S}$ do
        $r[i] \leftarrow (1 + w[i])q[i]$
    end for
    choose a site $m$ maximizing $r[m]$ and $d_{i,j}[m][j] \leq \tau$, where $j$ is the previously
    scheduled site
    $w2 \leftarrow w$
    for each site $S_i \in \mathcal{S}$ do
        $w2[i] \leftarrow w2[i] + 1$
    end for
    consider scheduling $S_m$
    $w2[m] \leftarrow 0$
    for each site $S_i \in \mathcal{S}$ do
        $r[i] \leftarrow (1 + w2[i])q[i]$
    end for
    choose a site $k$ maximizing $r[k]$ and $d_{i,j}[k][m] \leq \tau$, where $m$ is the previously
    considered scheduled site
    for each site $S_i \in \mathcal{S}$ do
        $r[i] \leftarrow (2 + w[i])q[i]$
    end for
    let $r[\ell] \leftarrow \max_i\{r[i]\}$
    for each site $S_i \in \mathcal{S}$ do
        $w[i] \leftarrow w[i] + 1$
    end for
    if $r[\ell] \geq r[m] + r[k]$ then
        schedule blank
    else
        schedule $m$
        $w[m] \leftarrow 0$;
    end if
    $t \leftarrow t + 1$
end for
```

Algorithm 3: Greedy Two-Steps Lookahead Algorithm.

consider scheduling the site m that can be scheduled from the previously scheduled site (note: we assume that in timeslot 0 all the sites have been visited and hence scheduled) within one timeslot, dictated by transition matrix $\mathcal{D}$ and that has the highest potential cost $r[m]$, which is calculated as $r[m] = (1 + w_2[m])q[m]$. For each i, increment $w_2[i]$ by one and set $w_2[m] = 0$. Next consider scheduling the site k that can be scheduled from the previously scheduled site m within one timeslot dictated by transition matrix $\mathcal{D}$ and that has the highest potential cost $r[k]$, which is calculated as $r[k] = (1 + w_2[k])q[k]$. Compare it with the potential cost if there is an empty slot followed by scheduling any site ℓ. In other words, pick the largest $r[\ell] = (2 + w[\ell]) \cdot q[\ell]$. Compare $r[\ell]$ with $r[m] + r[k]$., and pick the largest residual cost. If the largest is $r[\ell]$, schedule an empty slot and increment all $w[i]$; otherwise, schedule site m and increment all $w[i]$, for $i \in \{1, \ldots, n\} - \{m\}$, and make $w[m] = 0$.

```
let $t(\ell)$ be the subscript of the site scheduled at time $\ell$ for $1 \leq \ell \leq k$
assume we have scheduled sites $\mathcal{D} = S_{t(1)}, ..., S_{t(k)}$
among immediately schedulable sites, schedule a site $i$ maximizing $a_i + b_i(g_{i,t})$
```

Algorithm 4: Greedy $(a_i + b_i(g_{i,t}))$-based Algorithm.

```
for $s = 1$ to $T$ do
  run $([x_s], [y_s]) \leftarrow IP(s, L, [x_{s-1}], [y_{s-1}])$
  update column $s$ of $[x_{i,s}]$ matrix with array $[x_s]$ values, where $[x_s]$ is derived from IP
  solution
  update column $s$ of $[y_{i,s}]$ matrix with array $[y_s]$ values, where $[y_s]$ is derived from IP
  solution
  $s \leftarrow s + 1$
end for
Output $[x_{i,t}]$ matrix as a solution.
```

Algorithm 5: Greedy Chained-IPs: IP(L,T).

Other greedy algorithms do not calculate frequency array but use a_i and b_i costs directly.

```
let $t(\ell)$ be the subscript of the site scheduled at time $\ell$ for $1 \leq \ell \leq k$
assume we have scheduled sites $\mathcal{V} = S_{t(1)}, ..., S_{t(k)}$
next to schedule: site $S_{t(k+1)}$. Select it as follows:
let $w_i$ be associated gaps at time $k$
pick the next site to schedule with biggest $\frac{(1+w_i)(2+w_i)}{2} b_i + (1 + w_i)a_i$ that can be transitioned from site $S_{t(k)}$.
```

Algorithm 6: Greedy $(a_i + b_i(g_{i,t}))$-based Look-back Algorithm.

Algorithm 4 schedules sites as follows. At each step it greedily picks a site with maximum $a_i + b_i(g_{i,t})$. In other words, assume we already have a schedule up to time k which has a cost of $\mathcal{C} = \frac{1}{k} \sum_{i=1}^{n} \sum_{t=1}^{k} a_i(1 - x_{i,t}) + b_{i,t}(t - y_{i,t})$. Pick the next site i with maximum $C + (a_i + b_i(g_{i,t}))$ that can be transitioned from the previously scheduled site.

Another algorithm we implemented repeatedly solves IPs to construct a solution (see Algorithm 5).

Notation 2. *In the following algorithm 5, let*

- T *be the schedule period,*
- L *be the number of slots to look ahead,*
- $[x_{i,t}]$ *be an* $n \times T$ *matrix with* $0/1$ *entries indicating which site is scheduled at time* t *(each column* t *has exactly one* 1 *entry),*
- $[y_{i,t}]$ *be an* $n \times T$ *matrix that tells last appearances of all sites* i *at time* t*,*
- s *be a starting timeslot to run the IP,*
- $[x_p]$ *be a* $0/1$ *array of scheduled sites at time* $t = p$ *produced by IP,*
- $[y_p]$ *be an array of the times of last observing sites at time* $t = p$ *produced by IP,*

- $[x_0] \leftarrow \{0, \ldots, 0\}$ *(initial values for* $t = 0$*),* $[y_0] \leftarrow \{0, \ldots, 0\}$ *(initial values for* $t = 0$*), and*
- $IP(s, L, [x_p], [y_p])$ *be an IP that starts at timeslot* s *and period* L *with some initial values of* $[x_p]$ *and* $[y_p]$.

	Expr 1:		*Expr 2:*		*Expr 3:*		*Expr 4:*		*Expr 5:*		*Expr 6:*		*Expr 7:*	
site i \ *cost*	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i
A	1	1	16	1	1	16	4	4	4	4	16	16	16	16
B	4	4	9	4	4	9	16	16	9	9	1	1	4	4
C	9	9	4	9	9	4	9	9	16	16	4	4	1	1
D	16	16	1	16	16	1	1	1	1	1	9	9	9	9

Table 3.3: Seven cost scenarios for four sites.

The algorithm chains IPs as follows. It finds an optimal solution with a period L for timeslots 1 to L. It selects the first scheduled site, updates the $x_{i,1}$ and $y_{i,1}$ values, and advances a window by 1. It runs IP again to find an optimal solution with a period L for timeslots 2 to $L+1$ using the updated $x_{i,1}$ and $y_{i,1}$ values, updates $x_{i,2}$ and $y_{i,2}$ values, and advances a window by 1 again. It continues doing this until it gets values for all $x_{i,t}$ and $y_{i,t}$ for $1 \leq t \leq T$. This fully specifies the schedule. The number of IP programs that we run is T, each time advancing a window by one and recording the best first site that the IP on L timeslots picks. This algorithm can be thought of as the cost-based L-Steps Lookahead greedy algorithm. Algorithm 4 is a special case of algorithm 5 where $L = 1$, L is a lookahead parameter and T is a period of the schedule parameter.

Another cost-based greedy algorithm is the $(a_i + b_i(g_{i,t}))$-based Look-back (see Algorithm 6). Just as in Algorithm 4, it does not need to calculate frequencies. A drawback for Greedy $(a_i + b_i(g_{i,t}))$-based algorithm is that it assumes that previously scheduled sites are scheduled optimally and just selects the best next site based on the a_i and b_i costs. The $(a_i + b_i(g_{i,t}))$-based Look-back algorithm does not make this assumption. At each step it greedily picks a site with maximum residual cost without assuming that the previously scheduled sites were optimal. It uses gap information $g_{i,t}$ (which in our greedy solution we refer to by w_i with no reference to t since the schedule is built incrementally) to select the best site. In other words, assume we already have a schedule up to time k. At time $k + 1$, the algorithm picks a site maximizing $(w_i + 1)a_i + (1 + 2 + \cdots + w_i + (w_i + 1))b_i$, where w_i is a gap of site i at time k. Note that the fixed cost for a site of a schedule grows linearly, and variable cost for a site of a schedule grows quadratically with respect to the gap.

3.7 Testbed Architecture

Our experimental evaluation concerns the following application. A border, which may be a straight line or a curve, has n unprotected sites that need to be monitored. These n potential sites are distributed uniformly at random. A single sensor is positioned, hidden in front of the border, that is responsible in monitoring intrusions from these unprotected sites. We implemented the algorithms above to schedule a single sensor to visit n sites periodically while minimizing the potential loss of limited observations. The refocus delay time is proportional to the angle that the sensor needs to sweep to switch from one site to the next.

3.7.1 Various Experiments

We have conducted various experiments where a single sensor is scheduled to observe four sites; let's call them A, B, C, D respectively. Sites are located on a border that is 10 units

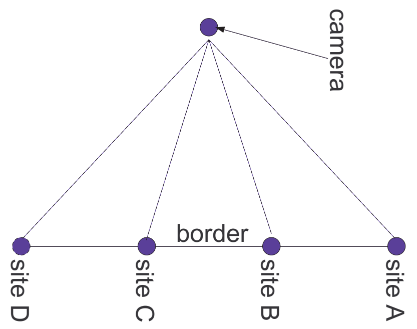


Figure 3.2: Experiment setup.

away from the sensor. The relative angles that each site makes with the other with respect to the sensor are summarized in the Table 3.4.

$Angle$	A	B	C	D
A	$0°$	$30°$	$60°$	$90°$
B	$30°$	$0°$	$30°$	$60°$
C	$60°$	$30°$	$0°$	$30°$
D	$90°$	$60°$	$30°$	$0°$

Table 3.4: Angles between sites.

We conducted various experiments where a single sensor is scheduled to observe four sites A, B, C, D, spaced equally apart, on a border located at distance 10 from the sensor (see Figure 3.2 and Table 3.4 for an illustration of this configuration and the resulting angles between cameras and sites, respectively). Our assumption is that a camera can visit the site and be able to sweep $45°$ within one time slot. If the next site is within more than $45°$ from the currently visiting site, then the camera cannot observe those two sites in immediate succession. In this example, the only adjacent sites can be reached and observed in the next timeslot.

We solved the IP using CPLEX to find optimal schedules for several different scenarios for the costs of the four sites (see Table 3.3), to investigate what patterns might emerge. In all seven experiments we have selected costs to be perfect squares for simplicity, due to the square-root law [11], which states that in an optimal schedule (absent delay constraints and with $a_i = a_j = 0$), the ratio of visit frequencies should be proportional to the square-root of this ratio. For example, if $b_i = 4$ and $b_j = 16$, then site j will be visited twice as often as site i, since $\frac{\sqrt{16}}{\sqrt{4}} = 2$.

	IP	One-Step $q_i = \frac{\sqrt{a_i+b_i}}{\sum_{i=1}^{n} \sqrt{a_i+b_i}}$	One-Step $q_i = \frac{1}{\tau_i}$	2-Step Look-ahead	$a_i + b_i(g_{i,t})$-based	$a_i + b_i(g_{i,t})$-based Look-back	IP(10, 21)	LB
Exper. 1	61.05	74.57	67.52	67.57	66.57	62.95	64.81	54.85
Exper. 2	68.05	78.29	76.24	72.43	81.52	78.29	69.71	59.88
Exper. 4	53.09	55.67	55.67	55.67	58.29	54.67	54.86	54.85
Exper. 5	57.62	59.14	58.76	58.76	66.57	58.57	60.86	54.85
Exper. 6	77.43	93.95	98.71	80.48	168.00	108.19	78.81	54.85
Exper. 7	75.67	90.38	92.71	84.81	140.19	103.95	78.95	54.85

Table 3.5: Comparison of IP and Greedy schedule costs per slot.

First we describe the nature of these cost value settings, as shown in Table 3.3. In experiment 1, fixed costs a_i and variable costs b_i increase for each additional site. In experiment 2, variable costs b_i increase for each additional site, but fixed costs decrease. Note, however, that the costs associated with b_i grow quadratically with the gap size, whereas the costs associated with the a_i grow linearly with the gap size. Experiment 3 reverses this, with variable costs b_i decreasing and fixed costs increasing. This setting is identical to the one in experiment 2 with the sites are renamed. In experiments 4 and 5, sites B and C have greater values for both costs, and are in the middle, surrounded by sites A and D. In experiments 6 and 7, the high-cost sites A and D are on the outside, surrounding sites B and C.

We solved the IP to obtain an optimal schedule (for period $T = 21$) for each of these cost settings. These optimal periodic schedules are shown in Figure 3.3. We also ran the

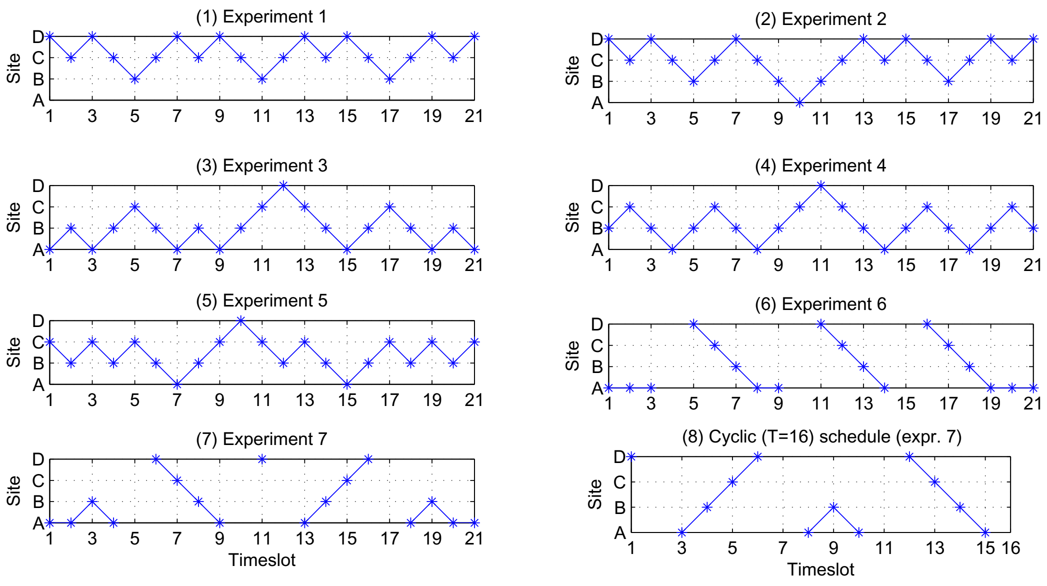


Figure 3.3: IP solutions for the 7 experiments.

heuristic algorithms above on these problem instances and compared the resulting schedule costs with the optimal costs from the IP, as well as a lower bound on the optimal cost. (see Table 3.5. The cost of the solution to the lower bound is obtained by solving Equation (3.4.2).

Next we conducted an experiment to compare the schedule costs of the different algorithms on randomly generated problem instances, where the *fixed* (i.e. a_i) and *variable* (i.e. b_i) costs are both chosen uniformly at random from the real interval $[0, 10]$. The number of sites n is ranges from 4 to 8, placed linearly along a border. As in the previous seven experiments, the adjacent sites can be transitioned between instantly, and non-adjacent sites can be transitioned between in one idle slot. For each such n, we average the results of 100 trials (see Figure 3.4).

We find that IP(L, T) performs quite well in all the experiments, even for small L.

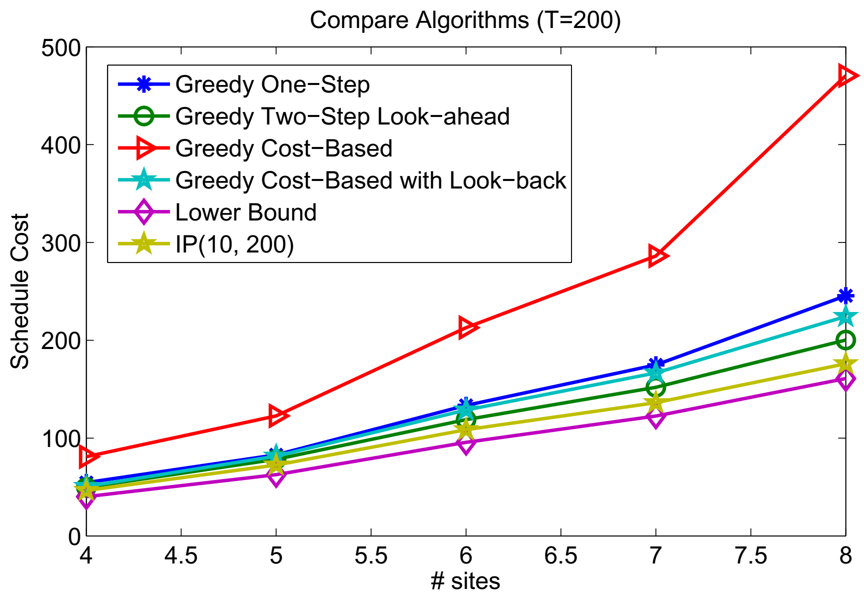
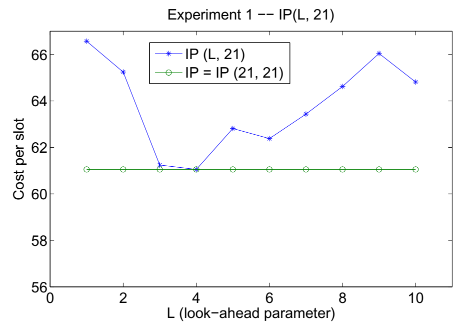


Figure 3.4: Comparison of Algorithms.

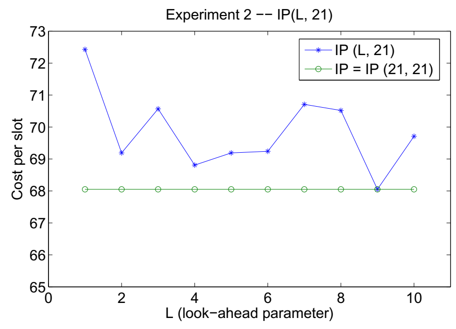
Of course, L should depend on the number of sites to be scheduled. For the lookahead parameter $L = 10$ the algorithm runs quite fast for any value of T. To investigate the influence of the value L ,we conducted experiments varying L with IP(L, T) solved for a fixed $T = 21$ (see Figure 3.5). Using what we found to be the best two values for L, we conducted another set of experiments on IP(L, T) where L is fixed and the period T is varied (see Figure 3.6).

3.7.2 Insights from Simulation

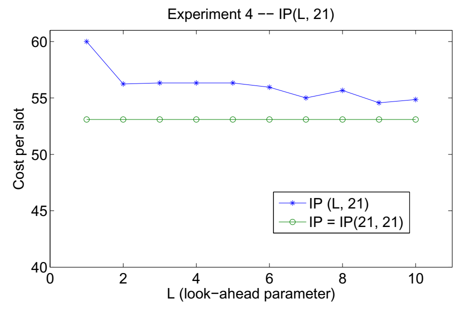
The schedules produced by IP for the seven experiments with sites A, B, C, D positioned along the border are depicted in Figure 3.3. Recall that in experiment 1 (see Figure 3.3(1)), the fixed and variable costs both increase for each site. Absent delay constraints, this would dictate that D be visited the most often, C the second most, and so on. Since site D is on the edge, instantaneously reachable from only C, D is actually visited less often than C



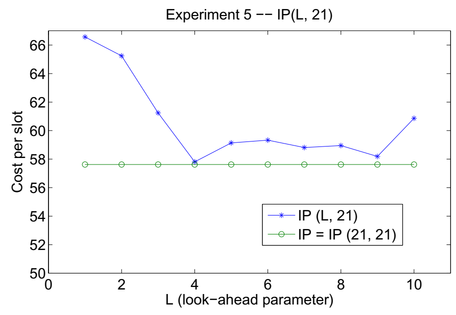
(a) Experiment 1.



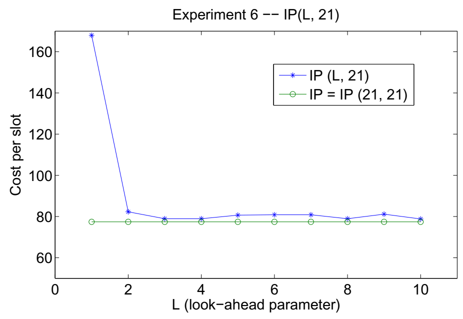
(b) Experiment 2.



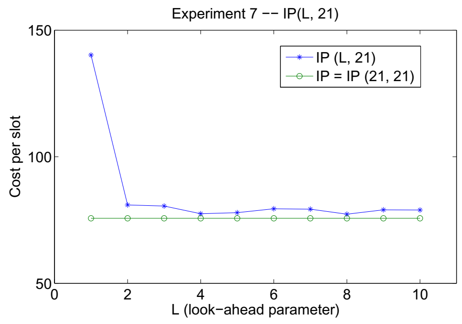
(c) Experiment 4.



(d) Experiment 5.

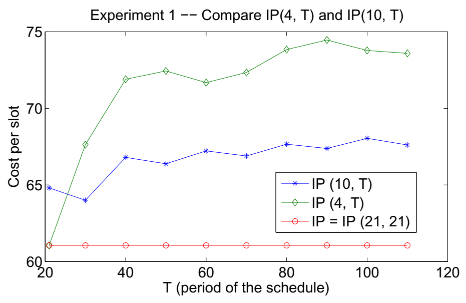


(e) Experiment 6.

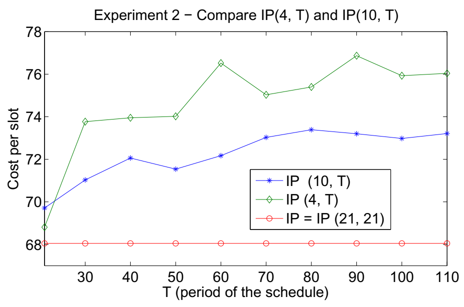


(f) Experiment 7.

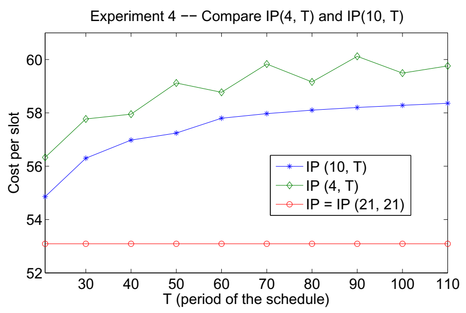
Figure 3.5: Compare solutions of IP(L, 21) by varying L.



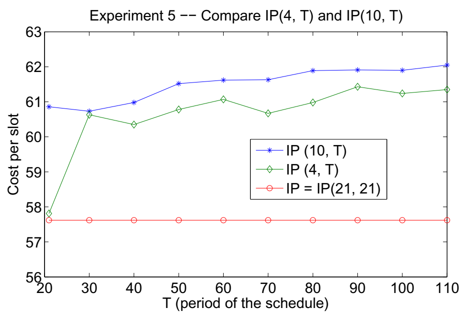
(a) Experiment 1.



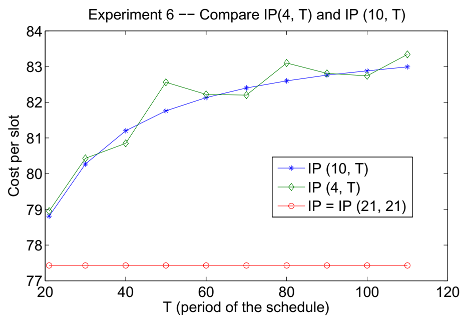
(b) Experiment 2.



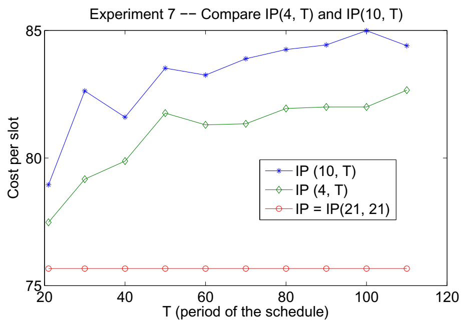
(c) Experiment 4.



(d) Experiment 5.



(e) Experiment 6.



(f) Experiment 7.

Figure 3.6: Compare solutions of IP(4, T) and IP(10, T) by varying T.

is. The refocus delays, together with site positions, play important roles in determining the character of the optimal schedule. (Site A was not scheduled at all, though it would be for sufficiently large period T.)

Recall that in experiments 2 and 3 (see Figure 3.3(2-3)), fixed cost grows as variable cost shrinks. What we find is that site A's high fixed cost results in it now being visited (with period still $T = 21$). Site C remains the most popular, however, due to its central position.

Recall that in experiments 4 and 5, the high-cost sites are in the middle, surrounded by the low-cost sites (see Figures 3.3(4) and 3.3(5)). In both experiments, site A has the same fixed and variable costs, but because A's neighbor B is the highest cost site in experiment 4, it is scheduled twice as often in that experiment as it is in experiment 5, in which B is only the second highest cost site. When fixed and variable costs are held fixed, relative positions of sites play a crucial role in determining the schedule.

Finally, recall that in experiments 6 and 7, the high-cost sites are on the outside, surrounding the low-cost sites (see Figures 3.3(6) and 3.3(7)). Now there is more incentive to schedule outside sites, but the problem is that the camera cannot switch instantaneously between these two important sites. Thus we find the optimal schedules using idle slots for refocusing.

It is interesting to note that in all 7 experiments the schedules appear to be cyclic. In fact, when we decrease the time horizon in experiment 7 to 16, we again obtain a periodic schedule (see Figure 3.3(8)).

From these seven experiments we observe that due to refocus parameter the frequency of a scheduled site in the optimal schedule does not only depend on the fixed and variable costs of a site, but on its relative position to other important sites in the region. In addition, it is sometimes optimal to have idle slots during which no site is scheduled but the entire time is taken by the sensor to move or refocus.

We also present results for the algorithms run on randomized problem instances (Figure 3.4). The period in these experiments is arbitrarily chosen to be $T = 200$. We find that Greedy Two-Steps Lookahead outperforms Greedy One-Step for frequency-based algorithms, and the cost-based Look-back Greedy outperforms the simple cost-based Greedy. IP(10, 200), significantly outperforms the others, and comes close to the IP optimal cost. We conducted an experiment where we run algorithms without delay constraints and observed that the costs of optimal schedules can come arbitrarily close to the cost of the lower bound schedule.

Since IP(10,21) comes close to the IP optimal solution, what about IP(L,21) with smaller values of L? Next we performed versions of the seven 4-site experiments with different L (see Figure 3.5, in which the IP optimal, which does not depend on a lookahead parameter, is plotted for comparison). What we find is that the curves given by IP(L, T) fluctuate. A common pattern is that as L increases, the IP(L, T) curve zigzags up and down. In Figures 3.5(e)-3.5(f) we see that the bigger the lookahead parameter L, the closer the IP(L, T) curve gets to the IP curve, but again with a fluctuating pattern that appears periodic.

We examine two particular lookahead values in our last set of experiments (see Figures 3.6). Here we plot IP(4, T) and IP(10, T), varying time horizon T. Even though for $T = 21$, the IP(4, T) curve is closer to the IP curve (as seen in Figures 3.6(a), 3.6(b), 3.6(d)), it jumps up for larger values of T. We also find that the curve for IP(10, T) is smoother than IP(4, T)'s in all settings, and lower for most of them. Even for those cases, the cost is nearly the same. We conclude that for longer time horizons, a larger lookahead value gives better results, as expected, and smoother behavior.

3.8 Conclusion and Future Work

In this chapter we have studied a scheduling problem of a single sensor observing n sites. We considered the time of refocus delay and its impact on scheduling. Our work poses interesting new problems. Since refocusing sensors from site to site may involve physically rotating or moving sensors, refocusing may consume a considerable amount of energy. Energy conservation may be of the essence. Scheduling sensors optimally to observe sites while at the same time conserving energy is an interesting open problem which we leave for our future work.

The single sensor scheduling problem can also be extended to a multiple-sensors setting. In the multiple sensor scheduling we want to schedule multiple sensors observing much bigger quantity of sites. One approach here would be to efficiently partition a set of sites into subsets and then assign each sensor its own subset. Another approach would be to have sensors cooperate by scheduling all m sensors to observe n sites. We defer such problems to future work.

Chapter 4

Throughput Maximization in Mobile WSN Scheduling with Power Control and Rate Selection

This chapter[1] presents a problem of data dissemination in mobile wireless sensor networks employed in applications such as rescue missions. The goal is to maximize the throughput of the data pulled by clients from the stationary access points, which can adaptively control their transmission rate and power.

4.1 Introduction and Motivation

In wireless sensor networks (WSNs), sensor nodes collect data of interesting events across the network and send them back to the data access points (APs), which are often stationary sensor nodes, awaiting the end users to collect the information on demand. It is often

[1]The work of this chapter is published in [9].

assumed in the literature that end users have immediate and unlimited access to APs via wired connections. However, if an end user is moving such that the wanted data needs to be **wirelessly** downloaded from an AP only when the user passes by, then the collection of data is subject to constrained contact windows in time. Furthermore, when there are more than one end users in the network to collect data from the given set of APs, they compete for the limited APs and constrained contact windows. In this case, how to assign the multiple APs to the mobile end users in time forms a job-machine scheduling problem with n jobs (each with weight w_i and processing time p_i, and release time r_i and deadline d_i) to be assigned to m parallel machines [24]. A valid assignment of a job i to machine k would be to dedicate machine k exclusively to job i over some interval $[s, s+p_i) \subseteq [r_i, d_i)$.

Systems for scenarios, like a rescue mission, where mobile end users need to download data items wirelessly from APs, introduce another degree of freedom to the scheduling problem, i.e., adaptive transmission rate selection. First assume APs can transmit using a constant transmission power or different channels that prevent any interference among neighboring APs. For a user-AP pair, the choice of transmission rate can affect both the contact window as well as the processing time, which in turn influences the scheduling performance. The reason is elaborated as follows.

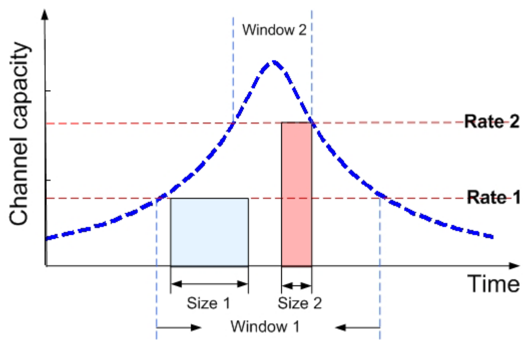


Figure 4.1: Time windows: various transmission rates and fixed power

According to the Shannon Theorem [52], (i.e. formula: $C = B \cdot \log\left(\frac{P/N}{d^2} + 1\right)$, where C - channel capacity, B - bandwidth of the channel, P/N - signal power to noise ratio, d - transmission range), as a user passes by an AP, the capacity of the channel from the AP to the user first increases, as the user approaches the AP, and then decreases, as the user departs the AP, as shown in Figure 4.1. The transmission rate of the download to the user is bounded above by the channel capacity of the AP. As a result, for a user-AP pair, choosing a lower transmission rate (i.e. Rate 1 in Figure 4.1) gives a larger contact window (i.e. Window 1 in the figure) with an AP as is seen by the flat Rate 1 line intersecting the capacity curve. Thus, lower transmission rates allow the download to start earlier and end later. For the higher rate (i.e. Rate 2 in Figure 4.1) the contact window (i.e. Window 2) is shorter which means the download should start later and should terminate earlier. Intuitively, the lower the transmission rate the larger the contact window size giving more freedom when the transmission can be started and finished. However, the other impact of rate control is on the download duration of job, i.e., the job size. Since we can transmit the job faster with high transmission rate the size of the job is shortened. On the other hand, with lower transmission rate the size of the job is longer. Thus, lower transmission rate requires more slots on an AP.

Therefore, selecting the transmission rate has a two-fold impact to the job-machine scheduling problem. Selecting a low transmission rate increases the job's contact window with an AP but at the same time increases the size of the job (i.e. its download duration), while selecting a high transmission rate decreases the job's contact window with an AP but at the same time decreases the size of the job. How to adaptively control the transmission rate to optimize the matching/scheduling between end users and APs is still an open issue, as existing rate control for WSNs mostly focus on resolving network congestions for data transmission from the source sensors to the APs [49] and [43].

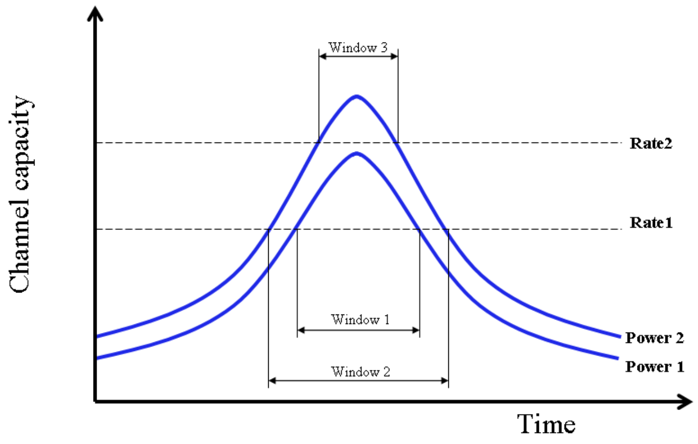


Figure 4.2: Time windows: various transmission rates and two different power levels

Controlling the transmission power adds another degree of freedom to our scheduling problem. When the power is increased the channel capacity curve would shift upward as is shown in Figure 4.2. On assumption that the transmission rate is fixed, using the formula $C = B \cdot \log\left(\frac{P/N}{d^2} + 1\right)$, we observe that increasing transmission power allows us to transmit farther by increasing APs transmission range. Thus, for a job-AP pair increasing power (i.e. from Power 1 to Power 2 in the Figure 4.2) increases the contact window (i.e. from Window 1 to Window 2 in the Figure 4.2), which means that the transmission can be started earlier and finished later. Even though power control with fixed transmission rate has no effect on the job's size, since job's size by definition depends on the size of the data and transmission rate, which intuitively means that we want to always use highest transmission power, it may create interference among other APs that are transmitting as

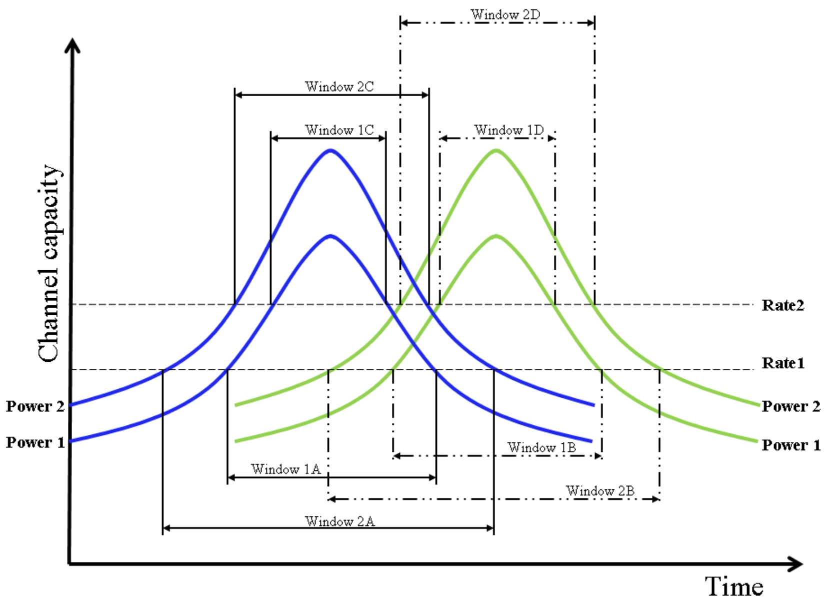


Figure 4.3: Time windows: various transmission rates and two different power levels and two interfering machines

is shown in Figure 4.3. In the figure we see that using Rate 2 and Power 2, Window 2C and Window 2D overlap (i.e. interference between two neighboring APs). However, when power is reduced to Power 1, Window 1C and Window 1D do not overlap. We also see from the figure that using lower transmission rate Rate 1 creates interference even for lower power Power 1. Thus, a balancing factor that prevents us from using the highest transmission power is interference.

To sum up, in the scheduling problem that we study in this paper both contact windows and job sizes depend on jobs, machines to which they are being assigned to, transmission power levels of APs, and transmission data rates at which data are being transmitted on a following machine to a following job. Both transmission rate and transmission power can be controlled. The goal is to schedule job transmission on APs so as to maximize the sum of profits of all scheduled jobs (i.e. throughput maximization) while controlling transmission rate and transmission power per each job-AP pair and at the same time eliminate interference among transmitting APs (e.g., only one AP can transmit to avoid interference while the neighboring APs need to reduce their transmission range, so as not to interfere, by controlling their transmission power and/or transmission rate).

The remainder of this chapter is organized as follows. Section 4.2 discusses some related work. Section 4.3.1 presents the system model. Section 4.3.2 and Section 4.3.3 present mathematical formulations of the problem and problem settings. Scheduling algorithms with rate selection and power control are presented in Section 4.3.4, followed by simulation and results in Section 4.4. Section 4.5 concludes the chapter.

4.2 Related Work

The scheduling problem here lies within the family of parallel-machine scheduling. The literature on scheduling algorithms on parallel machines is enormous [24]. The most abstract problem is the *Interval Scheduling Problem* or ISP which is formulated as follows. For $\forall i \in [1, n]$, given a family of intervals J_i. Selecting an interval [s, e) from J_i yields a profit of w_i. The task is to select at most one interval from each J_i so that the selected intervals are disjoint and the profit is maximized. This is the simplest model which is NP-hard [27]. The intervals may be listed explicitly or implicitly by some parameters defining a job J_i. A popular special case of ISP is where the intervals are defined by release time r_i, a deadline d_i, and a processing time p_i. To schedule job i, an interval of length p_i must be selected within the interval $[r_i, d_i)$. In the standard notation for scheduling problems this special ISP is equivalent to $1|r_i|\sum w_i(1-U_i)$. This problem is known to be NP-hard since a special case is a Knapsack problem when all deadlines are equal and all release times are 0. In fact this problem is NP-hard in the strong sense if different integer job lengths are allowed when all release times and deadlines are integers employing a simple reduction from 3-PARTITION [31] and [55]. To generalize ISP to multiple-machine case where the machines are unrelated the problem becomes $R|r_i|\sum w_i(1-U_i)$. Due to applications that these special cases of ISP solve, they are often referred to as *throughput maximization problem (TMP)* or *real time scheduling problem* [33], [21], [18], [25], and [27]. Many generalizations of this problem are NP-hard [55] when number of machines $m > 1$: $r_i = 0$ and identical d_i; three integer job lengths (1, 3, and q), integer deadlines but one overall release time; two integer job lengths (1 and q), integer release times and deadlines.

Many recent works on TMP provide approximation bounds for a more general setting of $R|r_i|\sum w_i(1-U_i)$ problem. Bar-Noy et al. in [18] give a 2-approximation for the $1|r_i|\sum w_i(1-U_i)$ and 3-approximation for the general case $R|r_i|\sum w_i(1-U_i)$ via an LP relaxation of a time-indexed formulation and rounding. They also provide a combinatorial algorithm m-Admission which has approximation bounds of $3+2\sqrt{2}$ for the unrelated machines case. Berman et al. in [21] give a combinatorial 2-approximation two-phase algorithm for $R|r_i|\sum w_i(1-U_i)$. Comparable results of 2-approximation are given by Bar-Noy et al. in [16] by employing a technique based on local-ratio which is comparable to primal-dual technique analysis. Chuzhoy et al. in [27] improve the approximation bound of 2 to less than 1.582 for arbitrary instances of ISP.

Many of these TMP algorithms considered machine independent contact windows [18], [21], and [16]. Recently, however, [25] has considered the TMP problem applied to mobile scenarios, where a mobile user can download from an AP only when it passes by within the AP's transmission range with machine-dependent contact windows. The problem is a generalization of TMP with job-dependant but machine-independent release times and deadlines. New algorithms with approximation guarantees are presented and evaluated. Lee et al. [40] study an unrelated machine scheduling where contact windows are both machine and job dependent. Their objective though is minimizing the total weighted flow time.

With an advance of wireless technology, wireless APs are capable of adjusting transmit power and data transmission rate with which an AP can communicate with the users [45] and [56]. Thus in wireless mobile applications there are other parameters that may specify/modify intervals and processing times for the TMP. The job scheduling problem relevant to adaptive rate-controlled scheduling for multimedia and other applications [61] and [42], is one in which each job $J_i = (w_i, r_i, d_i, p_{i,k})$ is instead characterized as $J_i = (w_i, r_i, \alpha_{i,k}, p_{i,k})$, where $\alpha_{i,k} = (d_i - r_i)/p_{i,k}$ is a stretch factor for J_i on machine M_k. Berman et al. in [20] presented a $2/(1 + 1/(2^{\lfloor\alpha\rfloor+1} - 2 - \lfloor\alpha\rfloor))$-approximation algorithm for this special case of TMP when the stretch factor α_i for each job J_i is at most α, which is a better than 2-approximation algorithm previously known. Though, the concept of a stretch factor is related to transmission rates, they are very different. In our application both jobs' processing times and contact windows depend on a transmission rate. Our multi-choice scheduling is related to a multiple-choice knapsack problem [41] in a sense where choices are rates that determine both contact window size of a job (i.e. $[r_i, d_i)$ intervals) and processing times p_i that are also machine-dependant.

The choice of power level determines the contact window size and hence performance of the schedule. Yang et al. in [56] consider a problem of throughput maximization in a wireless mesh access network where operating frequency and power levels can be adjusted. The problem is approached from a game theoretical perspective. In their work the goal is to maximize the SINR and hence the throughput of both cooperative and non-cooperative

APs while eliminating the interference. Peng et al. in [45] propose a recursive randomized algorithm to find optimal power levels and data rates for APs that would maximize the throughput. In both works, however, there is no scheduling involved since the objective is to transmit no matter to whom and at what time. As long as an AP can transmit some data with good SINR it contributes to the throughput. Our goal is to pick appropriate power level and data transmission rates for an AP for each job so as to eliminate interference with other APs as well as to maximize the schedule profit measured in sum of the weighted throughput of all scheduled jobs.

There are two models for the interference: physical and protocol. The *physical model* (e.g., SINR model) is widely considered as a reference model for physical layer behavior. However, its application in wireless sensor networks is limited due to its complexity. The *protocol model* (e.g., unified disk graph model) is simple. This is the model we use in our paper to create the interference matrix. Shi et al. in [53] reconcile the tension between physical and protocol models and explore the fundamental question on how to correctly use protocol interference model so as to narrow the solution gap between the physical and protocol models.

4.3 Problem Models and Algorithms

In this section we provide a formal problem definition and define an Integer Program (IP) to solve the problem. Since the problem is NP-hard, we then propose heuristic based algorithms with approximation guarantees to solve the problem.

4.3.1 Problem Models

We consider $\mathcal{M} = \{M_1, \ldots, M_m\}$ machines deployed in a given field and $\mathcal{J} = \{J_1, \ldots, J_n\}$ mobile users traveling in the field. Each user has a single job. Each of these jobs is associated with a profit, w_j. A user can be scheduled to download its job from any, but only one machine. Assume that both transmission data rates and transmission power levels are finitely discretized. A transmission rate out of $\mathcal{R} = \{R_1, \ldots, R_K\}$ pre-defined rate levels needs to be adopted for the download. Let $\mathcal{P} = \{P_1, \ldots, P_q\}$ be the set of discrete power levels which machines can select when transmitting data to each job.

With each pair of machines we can associate a zero-one interference matrix for each selectable power level and each selectable transmission data rate as $\mathcal{I}_{(M_k, M_\ell)}\,[\mathcal{P} \times \mathcal{P} \times \mathcal{R} \times \mathcal{R}]$. The entry is zero if two transmitting machines with selected power levels and selected transmission rates do not interfere, and one otherwise. That is, $\mathcal{I}_{(M_1, M_2)}\,[P_1, P_2, R_1, R_2] = 1$ means that M_1 selecting power P_1 and transmission rate R_1 would interfere with M_2 that selected power P_2 and transmission rate R_2.

We define the contact window associated with a chosen rate and power level of a job-machine pair to be the period of time within which the Shannon capacity [52] between the machine and the user is higher than the chosen rate (see Figure 4.1). Thus, release times and deadlines to download from the machines are job, machine, rate, and power dependent. The time it takes to download a job is the processing time that is also job, machine, and rate dependant. The processing times do not change for different power levels for a fixed transmission rate. The objective of the scheduling problem is then to find, for each job, a machine, a transmission rate, a transmission power and a set of consecutive timeslots (defined by a starting timeslot), so as to maximize the total scheduled job profit while at the same time making sure that the transmission of any job by one machine does not interfere

Table 4.1: IP formulation with adjustable Rates and Powers.

$$\max \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{\rho=1}^{K}\sum_{\pi=1}^{q}\sum_{s=r_{jk\rho\pi}}^{d_{jk\rho\pi}-p_{jk\rho}} w_j{\cdot}x_{jk\rho\pi s}$$

$$\text{s.t.} \sum_{j=1}^{n}\sum_{\rho=1}^{K}\sum_{\pi=1}^{q}\sum_{u=s-p_{jk\rho}+1}^{s} x_{jk\rho\pi u} \leq 1 \qquad \forall_{k,s} \qquad (4.3.1)$$

$$\sum_{k=1}^{m}\sum_{\rho=1}^{K}\sum_{\pi=1}^{q}\sum_{s=r_{jk\rho\pi}}^{d_{jk\rho\pi}-p_{jk\rho}} x_{jk\rho\pi s} \leq 1 \qquad \forall_{j} \qquad (4.3.2)$$

$$x_{jk\rho_1\pi_1 s} + \sum_{u=s-p_{i\ell\rho_2}+1}^{s} x_{i\ell\rho_2\pi_2 u} +$$

$$+\mathcal{I}_{k,\ell}(\pi_1,\pi_2,\rho_1,\rho_2) \leq 2 \qquad \forall_{k,\ell}(k \neq \ell) \qquad (4.3.3)$$

$$\forall_{i,j}(i \neq j)$$

$$\forall_{\pi_1,\pi_2,\rho_1,\rho_2}$$

$$\forall_{s}$$

$$x_{jk\rho\pi s} \in \{0,1\} \qquad (4.3.4)$$

with the transmission of jobs from any other machine when powers and rates are adjusted.

We use indices $i, j \in \{1, \ldots, n\}$ for jobs, $k, \ell \in \{1, \ldots, m\}$ for machines, $\rho, \rho_1, \rho_2 \in \{1, \ldots, K\}$ for rates, $\pi, \pi_1, \pi_2 \in \{1, \ldots, q\}$ for power levels, and $s \in \{1, \ldots, t\}$ for timeslots. We can then express release time and deadline for job j on machine k with transmission rate ρ and power level π as $r_{jk\rho\pi}$ and $d_{jk\rho\pi}$. The processing time for job j, machine k, and transmission rate ρ is $p_{jk\rho}$. Let s indicate the starting time of job j on machine k with transmission rate ρ and transmission power π if this job assignment (job instance) is chosen. In such a case $r_{jk\rho\pi} \leq s$ and $s + p_{jk\rho} \leq d_{jk\rho\pi}$.

4.3.2 IP Formulation

The problem can be formulated as an Integer Program (IP). Even though in general solving an IP is an NP-hard problem, for moderate input sizes of the problem instances an IP can give a solution in reasonable time. For larger instances LP-relaxation may provide useful bounds to compare heuristics.

We extend an IP formulation of [25] to one where multiple transmission rates and powers are possible. Let $x_{jk\rho\pi s}$ be a 0/1 variable for each job instance $[s, s + p_{jk\rho})$ of J_j on M_k with transmission rate $\rho \in \mathcal{R}$ and that uses a transmission power $\pi \in \mathcal{P}$. It is 1 if job j is scheduled on machine k with transmission rate ρ, power level π, starting at timeslot s. Otherwise it is 0. The IP formulation is shown in Table 4.1.

The objective here is to maximize the sum of weight of all scheduled jobs. The first constraint ensures mutual exclusion, that is, no multiple jobs are scheduled simultaneously on a single machine. The second constraint prevents any single job from being scheduled more than once. The third constraint ensures that there is no interference among transmitting APs. The fourth constraint restricts x variables to be $0/1$ integer. For an LP-relaxation, we can relax this constraint to be any number in interval $[0, 1]$.

To study how power control affects the scheduling performance independently from rate control, we can assume that there is only a single transmission rate that the APs may employ and model the problem for adjustable power levels only. In this case both release times and deadlines are job-machine-power dependant as $r_{jk\pi}$ and $d_{jk\pi}$, but the processing times are job-machine dependent only as in p_{jk}. The interference matrix then is modified as $\mathcal{I}_{(M_k,M_\ell)}\,[\mathcal{P} \times \mathcal{P}]$. The entry is zero, if two transmitting machines do not interfere at given transmitting power levels, and one otherwise. The x variables will drop ρ subscript since the rate is fixed. The IP can be modified by dropping the third summation from

Table 4.2: IP formulation with adjustable Powers.

$$\max \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{\pi=1}^{q}\sum_{s=r_{jk\pi}}^{d_{jk\pi}-p_{jk}} w_j \cdot x_{jk\pi s}$$

$$\text{s.t.} \sum_{j=1}^{n}\sum_{\pi=1}^{q}\sum_{u=s-p_{jk}+1}^{s} x_{jk\pi u} \leq 1 \qquad \forall_{k,s} \qquad (4.3.5)$$

$$\sum_{k=1}^{m}\sum_{\pi=1}^{q}\sum_{s=r_{jk\pi}}^{d_{jk\pi}-p_{jk}} x_{jk\pi s} \leq 1 \qquad \forall_{j} \qquad (4.3.6)$$

$$x_{jk\pi_1 s} + \sum_{u=s-p_{i\ell}+1}^{s} x_{i\ell\pi_2 u} + + \mathcal{I}_{k,\ell}(\pi_1,\pi_2) \leq 2 \qquad \forall_{k,\ell}(k \neq \ell) \; \forall_{i,j}(i \neq j) \; \forall_{\pi_1,\pi_2} \; \forall_s \qquad (4.3.7)$$

$$x_{jk\pi s} \in \{0,1\} \qquad (4.3.8)$$

the objective and the second summation from both the first and second constraints. The formulation is given in Table 4.2.

The constraints are as before. The first constraint ensures mutual exclusion, that is no multiple jobs are scheduled simultaneously on a single machine. The second constraint prevents any single job from being scheduled more than once. The third constraint ensures that there is no interference among transmitting APs. The fourth constraint restricts x variables to be $0/1$ integer.

When APs are located in such a way that no interference is possible for the highest selected power level for each AP we can eliminate the interference constraint from the formulation altogether. The release times, deadlines, and processing times in this case

Table 4.3: IP Formulation with adjustable Rates.

$$\max \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{\rho=1}^{K}\sum_{s=r_{jk\rho}}^{d_{jk\rho}-p_{jk\rho}} w_j \cdot x_{jk\rho s}$$

$$\text{s.t.} \sum_{j=1}^{n}\sum_{\rho=1}^{K}\sum_{u=s-p_{jk\rho}+1}^{s} x_{jk\rho u} \leq 1 \qquad \forall_{k,s} \tag{4.3.9}$$

$$\sum_{k=1}^{m}\sum_{\rho=1}^{K}\sum_{s=r_{jk\rho}}^{d_{jk\rho}-p_{jk\rho}} x_{jk\rho s} \leq 1 \qquad \forall_{j} \tag{4.3.10}$$

$$x_{jk\rho s} \in \{0,1\} \tag{4.3.11}$$

become job-machine-rate dependent only and are expressed as $r_{jk\rho}, d_{jk\rho}, p_{jk\rho}$ respectively. The x variables will drop the π subscript since the powers are fixed. This can help us study how adjustable rates affect the scheduling performance. The IP formulation is modified by dropping the fourth summation from the objective and the third summation from both the first and second constraints. The formulation is given in Table 4.3.

The first set of constraints is for mutual exclusion which prevents multiple jobs from being scheduled on a single machine. The second set of constraint prevents each job from being scheduled more than once. The third constraint restricts decision variable to be integers 0 or 1.

We have implemented these IPs in AMPL and solved them with CPLEX. For large instances we can relax integrality constraints to solve an LP-relaxation of IPs.

4.3.3 Problem Setting

In this section we describe our environment and discuss two different settings on which we base our solutions. In the system we have stationary Access Points deployed within a geographical region and mobile clients with information needs that are traveling towards a mission site. We call the time period during which a client can communicate with the AP the *contact window*. A client must receive its information from any one of the AP within this time window. The window is decided by the speed vector, the route, the relative location of the AP, the transmission rate and transmission power of the AP.

The system has slotted time, so the task of the system is to decide how we allocate these timeslots to different clients. There are two types of problem settings we consider: offline and centralized online. In the offline setting, everything required to solve the problem is known, i.e., the data that is requested, the path of each mobile client including their speed. This setting is relevant for the scheduling a planned rescue missions. All scheduling may be done before the missions start. In the centralized online, nothing about the job is known until its client appears on a geographical region. Once the client is in the region, everything about the job becomes known, provided that nothing unpredictable happens. For example data delivery to busses in the city, where busses may be added or removed from service, but their routes are known and fixed. A central system has information about the busses in service and schedules the delivery of data. In the next section we design algorithms for both settings and prove approximation bounds.

4.3.4 Algorithms

In this section we design algorithms for the TMP($\mathcal{J}$,$\mathcal{M}$, $\mathcal{R}$, $\mathcal{P}$) problem. First we prove the hardness of our scheduling problem. We prove hardness by means of a Cook reduction from the Knapsack, which is NP-hard [31].

Theorem 4.3.1. *Solving TMP($\mathcal{J}$,$\mathcal{M}$, $\mathcal{R}$, $\mathcal{P}$) optimally is NP-hard.*

Proof. Given an instance of a Knapsack with capacity B and knapsack items i where $i \in$

$[1, n] \subset \mathbb{N}$. Each item i has a profit w_i and a size p_i. We create an instance of TMP($\mathcal{J}$, $\mathcal{M}$, $\mathcal{R}$, $\mathcal{P}$) where there is only one machine, one transmission rate and one power level (i.e., TMP($\mathcal{J}$, M_1, R_1, P_1)). For each knapsack item i associate a J_i of TMP($\mathcal{J}$,M_1, R_1, P_1), where all J_i have release time equal 0 and deadline equal B. The processing time of J_i is equal to the size of the knapsack item i, which is p_i. The weight of J_i is equal to the profit of knapsack item i, which is w_i. An optimal solution to TMP($\mathcal{J}$,M_1, R_1, P_1) is an optimal solution to Knapsack. $\square$

Since the general problem is NP-hard even in a restricted setting where there is only one choice for a transmission rate and transmission power, we propose heuristic based algorithms by extending existing combinatorial algorithms which in some cases preserve the approximation guarantees. We adapt Admission algorithm of [18] and Two-Phase algorithm of [21] to design our algorithm to solve machine-job-rate and machine-job-rate-power dependent scheduling problems. We note that both of these algorithms were adopted in machine-job dependent scheduling windows settings in [25] without losing approximation guarantees.

Admission based algorithms

In an Admission algorithm [18], jobs are considered in the order of non-decreasing end times. The algorithm schedules jobs machine-by-machine m times, and hence an algorithm is called m-Admission. The approximation ratio of this algorithm is $3 + 2\sqrt{2}$.

First we design algorithms that assume that transmission powers are fixed and no interference occur when adapting different transmission rates.

m-Admission – all rates algorithm We propose an "m-Admission – all rates" algorithm. (Refer to Algorithm 7), which is a straightforward extension of the "m-Admission" algorithm in [25] where all possible rate levels are considered. The algorithm proceeds as follows:

- For each job-AP combination (j, k), choose a rate ρ. Find out the contact window size $T_\rho = (d_{jk\rho} - r_{jk\rho})$ timeslots and job size $p_{jk\rho}$ timeslots associated with the chosen rate ρ. Then, enumerate $N_{jk\rho} = T_{jk\rho} - p_{jk\rho} + 1$ job instances, each with incremental starting timeslot s where $s \in [r_{jk\rho}, d_{jk\rho} - p_{jk\rho})$.
- Perform step 1 with all combinations job-machine-rate triplets (j, k, ρ) until all job instances (j, k, ρ, s) are enumerated.
- Run m-Admission on all the job instances to obtain the final schedule.

It is worth noting that "m-Admission – all rates" provides the same approximation guarantee (i.e., $3 + 2\sqrt{2}$) as the m-Admission algorithm of [18], since all possible rate levels are considered here.

m-Admission – max-ratio rate selection algorithm In systems with large number of possible rate levels, the algorithm "m-Admission – all rates" that enumerates job instances with all possible rate levels could be too complex. One way to avoid this is to find an existing one rate for each job-AP pair that can be considered rather than all possible rates. Our heuristic selects a rate so that the ratio of contact window size to the job size is maximized. More formally:

- For each job-AP combination (j, k), choose a rate $\rho^*_{j,k}$ such that $\rho^*_{j,k} = \arg_\rho \max \frac{d_{jk\rho} - r_{jk\rho}}{p_{jk\rho}}$.
- Each job-AP pair has a chosen rate $\rho^*_{j,k}$.

- Enumerate all job instances with the chosen rate with all different combination of $(j, k, \rho^*_{j,k}, s)$.
- Run m-Admission on the job instances to obtain the final schedule.

The algorithm will run faster on expense of losing the approximation guarantee.

Centralized-Online Algorithm The m-Admission algorithm can be extended to the centralized online setting. Rather than applying algorithm machine-by-machine, we can extend it so that the algorithm works on machines in parallel where we schedule the earliest finishing job among all the machines in each step. The extended algorithm is called *Global Admission* (Refer to Algorithm 8). To implement our centralized algorithm, we apply Global-Admission algorithm iteratively (Refer to Algorithm 9).

Two-Phase based algorithms

Just like with the m-Admission algorithm, we can adopt a Two-Phase algorithm of [21] which guarantees a slightly better performance. In the first phase the algorithm pushes job instances in order of non-decreasing end times onto a stack, assuming that these job instances have great enough weight compared to conflicting instances already on the stack. In the second phase, the algorithm pops the job instances from the stack to place them into a non-overlapping schedule. When a job instance enters the stack, it is pushed with a positive difference of its weight and the sum of all the weights of the overlapping instances on the stack. This in effect guarantees that the weight of the stack is equal to the weight of the schedule formed in the second phase. Just like with the m-Admission algorithm, we can enumerate all instances with all combinations of job-machine-rate triplets (j, k, ρ) until all job instances (j, k, ρ, s) are enumerated, where $s \in [r_{jk\rho}, d_{jk\rho} - p_{jk\rho})$. Then run the Two-Phase algorithm with all the instances. It is worth noting that the approximation ratio of 2 still holds in this case. We call this algorithm “Two-Phase – all rates”. We can further extend the algorithm where the rates are pre-selected based on max-ratio of contact window and job size. Afterwards, rather than running an m-Admission, we run the Two-Phase algorithm. We call this algorithm “Two-Phase – max-ratio rate selection”. Refer to the pseudo-code of the Two-Phase Algorithm in [21].

Two-Phase Algorithm with Controllable Power and Rate Levels Algorithm The Two-Phase algorithm can also be extended for the case with adaptive power control and transmission rate. Both rate and power levels are controllable. We must adapt different power levels and transmission rates ensuring that there is no interference between transmitting APs. The extended algorithm is called *Two Phase Algorithm – Rate-Power-Control (2PA-RPC)* (Refer to Algorithm 10).

Let *weight* be defined as the importance of the job. Let *value* be defined as the importance value that a job instance is pushed to the stack with, which equals the marginal value of scheduling this job instance, over the value of the job instances lower on the stack doing so would prevent us from scheduling (both those conflicting with this instance and other instances of the same job).

Algorithm 10 is the generalization of Berman’s Two Phase Algorithm to k (sometimes interfering) machines that works as follows: in phase one, job instances (for all machines) are considered in order of nondecreasing ending and pushed onto the stack if their (net) value is greater than that of the job instances there they would preclude scheduling. In phase two, instances are popped off of the stack and scheduled, with the instances they preclude popped off and thrown away.

A job instance is now specified by job, [start,end) interval, and machine. Machines can run at different power levels transmitting with different transmission rates (both corresponding to speeds), which determine the time a job will take. We assume job, interval length, and machine determine the power level and transmission rate that will yield this interval length, and so do not specify it. Two machines may interfere, however, depending on their locations, their power levels, and their transmission rates. That is, the entities that interfere with one another in this problem setting are pairs of (machine, power level, rate) triplets. (This generalizes a simpler, special case model on which pairs of machines interfere.) Two (machine, power, rate) pairs with the same machine in each pair *always* interfere.

A parameter appearing in the approximation guarantee is $\mathcal{I}$, which is the maximum number of mutually non-interfering machines that may simultaneously interfere with a single other machine. That is, imagine the interference graph between machines resulting from a power assignment to every speed. $\mathcal{I} + 1$ is the size of the largest claw (i.e., $K_{1,\mathcal{I}}$) contained in any such interference graph (for a given problem instance). A value yielding a looser approximation guarantee is the largest degree of all such graphs. Of course, $\mathcal{I} = 0$ when there is no interference between machines.

Consider job instance x defined as a tuple $(i, v, d, e, k, \pi, \rho)$. The entries of a given job instance x are indicated by $x.d$, $x.e$, etc. We say job instance x *conflicts with* y if $y.d < x.e \leq y.e$ (or vice versa) and $(x.k, x.\pi, x.\rho)$ interferes with $(y.k, y.\pi, y.\rho)$.

Theorem 4.3.2. *The approximation ratio of 2PA-RPC for problem TMP($\mathcal{J}, \mathcal{M}, \mathcal{R}, \mathcal{P}$) is at most* $2 + \mathcal{I}$.

Proof. We prove the result by extending the arguments of [21] from the single-machine case. Let $V(X)$ be the sum of values of intervals in the set of intervals X. The proofs of Lemmata 1 and 3 of [21], which prove respectively that the algorithm produces a valid, conflict-free solution and that the solution value (i.e., the total weight of the jobs scheduled in Phase Two) is at least $V(S)$, go through essentially unchanged. Our argument here follows and generalizes the proof of [21]'s Lemma 2.

By S we will refer both to the stack and the set of entries on it at the end of Phase One, when the algorithm is run on some fixed problem instance. Let O be some optimal solution (of value OPT).

Let S_x be the set of all non-$x.i$ job instances on S conflicting with x, and let $S_{x.i}$ be the set of all $x.i$ job instances on S.

Consider a particular job instance $x = (i, v, d, e, k, \pi, \rho) \in O$. S_x contains the non-$x.i$ job instances on the stack that would have conflicted with x at the time it was considered and possibly pushed onto the stack. (Recall that job instances are considered in order of ending.) Now consider $\sum_{x \in O} V(S_x)$. Each time a job instance $x \in O$ is considered and some instance $y \in S$ conflicts with it, the value of y is added to this sum. How many times can an instance y be counted? That is, for a particular y on S, how many intervals $x \in O$ can conflict with it? In the single-machine setting of [21], the intervals $[d, e)$ appearing in O are disjoint, and so the intervals on S each intersect with at most one interval of O—the intervals of O *partition* (a subset of) the intervals on the stack, and the answer is 1. In our setting, however, things are more complicated.

Because all the intervals $[d, e)$ appearing in O *for a given machine* k are disjoint, y's interval will intersect at most one member of O per machine. Less loosely, since $x.d < y.e \leq x.e$ in such cases, y conflicts with each such instance x at time step $y.e$ and yet the instances x do *not* conflict with one another, and so the largest possible number of such instances x for machines *other than* $x.i$ is $\mathcal{I}$. Of course, y could conflict with a (lower power) $y \in O$ on the same machine as y as well. Therefore:

$$\sum_{x \in O} V(S_x) \leq (1+\mathcal{I})V(S) \tag{4.3.12}$$

Consider again job instances $x \in O$ and the corresponding $S_{x.i}$. Since these are disjoint subsets of S, we have:

$$\sum_{x \in O} V(S_{x.i}) \leq V(S) \tag{4.3.13}$$

We now prove that for each $x \in O$, we have:

$$w_i \leq V(S_x) + V(S_{x.i}) \tag{4.3.14}$$

Let $\hat{S}$ be the set of job instances on the stack when $x = (i, v, d, e, k, \pi, \rho)$ is considered in Phase One. At that point we decide whether to push to the stack based on the value $v = w_i - TOT(x) - tot(i)$. By definition, we have $tot(i) = V(\hat{S}_{x.i})$ and $TOT(x) = V(\hat{S}_x)$. Therefore, if $v \leq 0$, we have:

$$\begin{aligned} w_i &\leq tot(i) + TOT(x) \\ &= V(\hat{S}_{x.i}) + V(\hat{S}_x) \\ &\leq V(S_{x.i}) + V(S_x) \end{aligned}$$

On the other hand, if $v > 0$ then we push (increasing the stack value by v), and so we have:

$$\begin{aligned} V(S_i) &\geq V(\hat{S}_{x.i}) + v \\ &= V(\hat{S}_{x.i}) + w_i - TOT(x) - tot(i) \\ &= V(\hat{S}_{x.i}) + w_i - V(\hat{S}_x) - V(\hat{S}_{x.i}) \\ &= w_i - V(\hat{S}_x) \\ &\geq w_i - V(S_x) \end{aligned}$$

Thus Ineq. 4.3.14 again follows.

Now we sum the LHS and the two terms of the RHS of Ineq. 4.3.14 over all $x \in O$ and apply Ineqs. 4.3.12 and 4.3.13:

$$\begin{aligned}
\sum_x w_i &\leq \sum_x V(S_x) + \sum_x V(S_{x.i}) \\
OPT &\leq (1+\mathcal{I})V(S) + V(S) \\
&\leq (2+\mathcal{I})V(S)
\end{aligned}$$

$\square$

4.4 Simulation and Results

We have conducted experiments on simulated data. Different scenarios are considered. In particular, in one set of experiments we have considered one AP deployed and have analyzed separately the effects of power control and rate selection. In another set of experiments we have considered APs deployed on a straight line and a convoy of clients (e.g., cars) traveling through APs along a straight line with varying speeds. In yet another set of experiments we have considered APs deployed on a grid. The density of the APs network is controlled by varying the d which is defined to be the distance between horizontal or vertical lines of a grid with each AP located on an intersection of horizonal and vertical lines of a grid. To generate jobs, we use the Random Waypoint model described in [23] and used in [25]. In this model a job (car) selects a random destination in the simulated region and a random speed in the range of [5, 10] m/sec. The weight of a job is randomly generated using a Zipf distribution with $\alpha = 2$, clipped with a minimum and maximum weights of 1 and 10 respectively. The data sizes are uniformly distributed in [1, 10]. The algorithms can adaptively choose rates from 1 to 11 Mbps ($K = 11$) in discrete steps of

1 Mbps. The bandwidth of APs is set to 20Mbps. The power levels can be controlled and adaptively chosen from a set $\mathcal{P} = \{100, 125, 150, 175, 200\}$. For each job-AP pair the contact windows (which are related to transmission ranges) are calculated for different transmission rates and different transmission powers using formula:

$$C = B \cdot \log\left(\frac{P/1}{d^2} + 1\right)$$

The interference matrix $\mathcal{I}_{(M_k, M_\ell)}\left[\mathcal{P} \times \mathcal{P} \times \mathcal{R} \times \mathcal{R}\right]$ is calculated using the protocol interference model. The entry is zero if the disks corresponding to transmission ranges of APs with radii given by the above formula, do not overlap for given transmission power levels and transmission data rates. The entry is one otherwise.

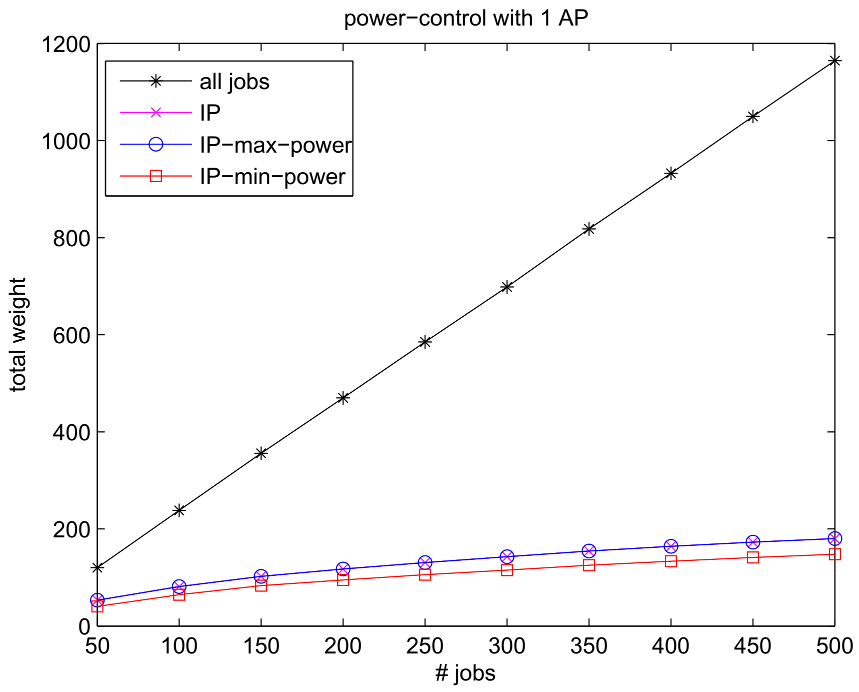


Figure 4.4: Power-Controlled throughput with 1 AP

In the first experiment depicted in Figure 4.4 we consider one AP and vary the number of jobs. The power levels can be controlled and chosen from a set $\mathcal{P} = \{100, 125, 150, 175, 200\}$.

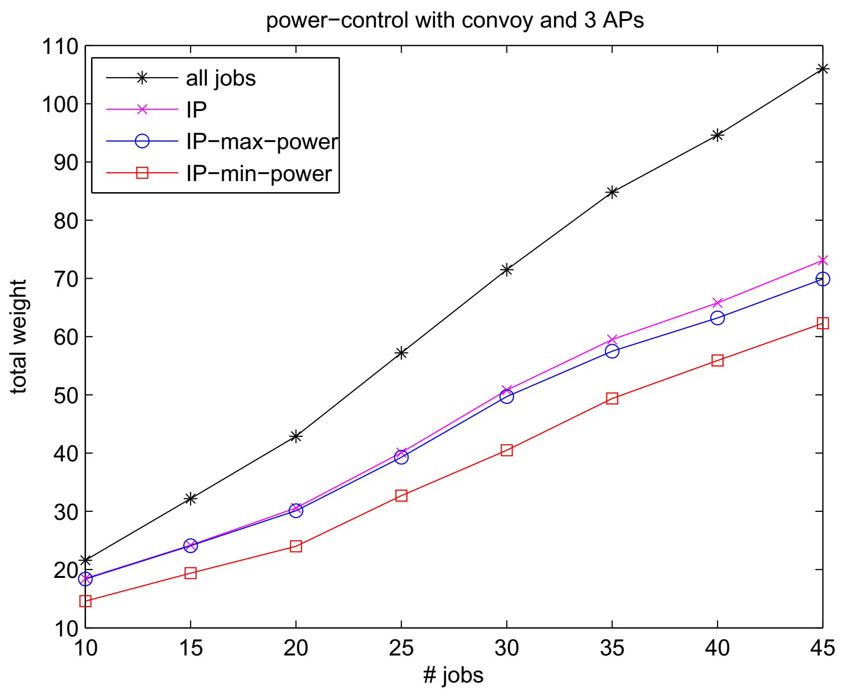


Figure 4.5: Power-Controlled throughput for convoy and 3 APs

The transmission rate is fixed at $1Mbps$. We run the IP with all possible power levels, with only maximum power level of 200, and with only minimum power level of 100. Figures 4.5 and 4.6 depict an experiment of power control in a scenario of three APs deployed on a line with distance of separation between APs equal to $110m$ and a convoy of cars traveling in the same direction with different speeds. Just like in the previous experiment power levels can be controlled while the transmission rate is fixed at $1Mbps$. The distance of separation, $d = 110m$, between APs is chosen so that there is no interference if neighboring APs both transmit with minimum power and interfere otherwise. In the next experiment we investigate effects of rate control. Figure 4.7 depicts an experiment with one AP with a fixed transmission power but adjustable rates chosen from a set $\mathcal{R} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. We run IP with all rate levels, with maximum rate level, with minimum rate level, and with

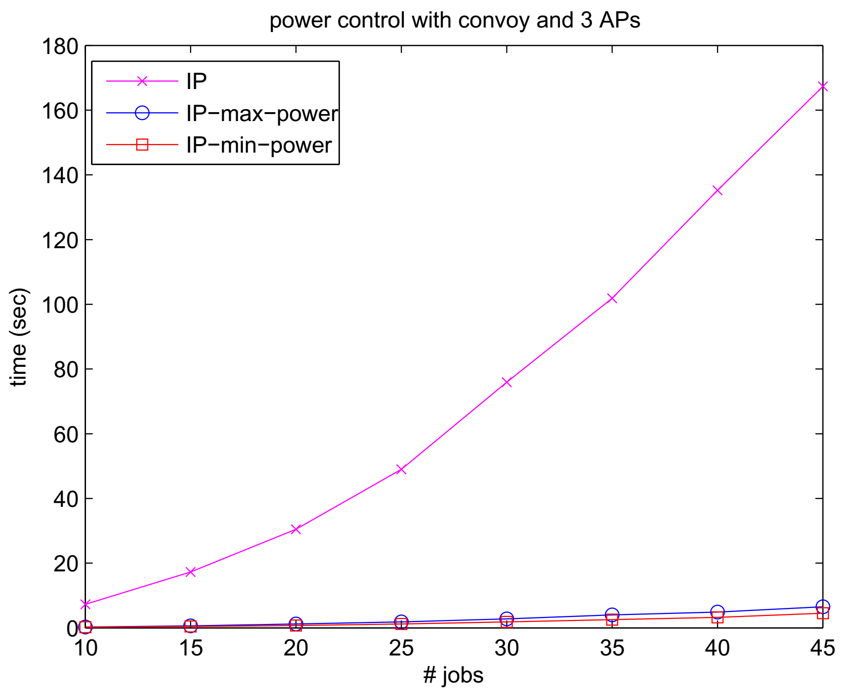


Figure 4.6: Runtime for convoy and 3 APs

rate level for each job-AP pair selected in such a way so that the ratio of contact window size over job size is maximized.

Next we have conducted experiments to test the performance of our algorithms for both offline and online settings. Figure 4.8 and Figure 4.9 depict experiments with a convoy passing through 9 APs on a line with $d = 90m$ and $d = 30m$, respectively. Figure 4.10 and Figure 4.11 depict experiments where 9 APs are placed on a grid of 3 rows and 3 columns with distance of separation of rows and columns $d = 90m$ and $d = 30m$ respectively. The clients are traveling using the Random Waypoint model. Algorithms evaluated are m-Admission, two-phase, centralized-online, both with all rates and preselected best rate based on maximum ratio of contact window size to job size.

Next we have conducted experiments to test the 2PA-RPC algorithm for the case of

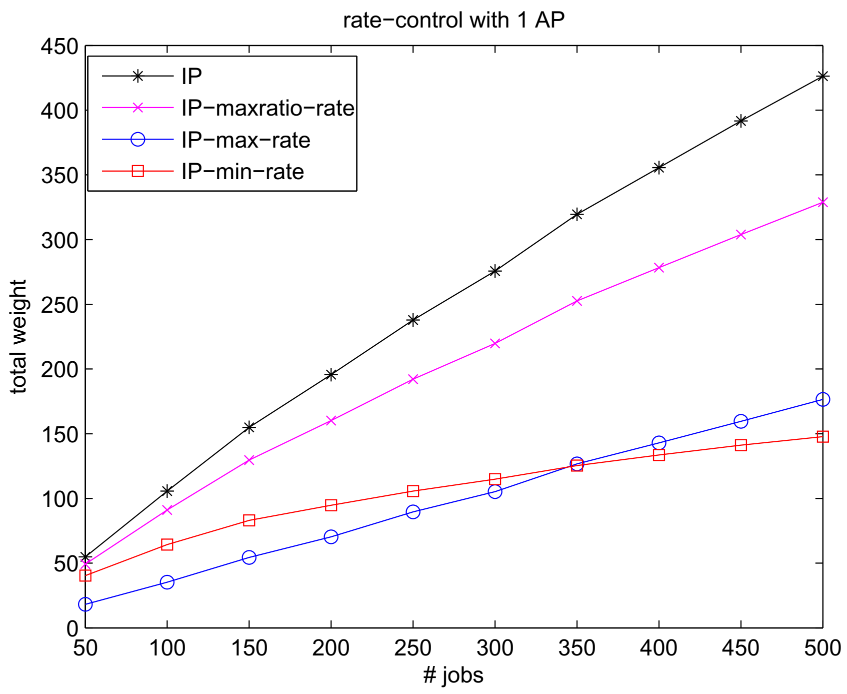


Figure 4.7: Rate-Controlled throughput with 1 AP

adjustable powers. Figure 4.12 depicts an experiment with one AP with a fixed transmission rate but adjustable powers chosen from a set $\mathcal{P} = \{100, 125, 150, 175, 200\}$ while Figure 4.13 depicts a convoy passing through 3 APs on a line. The distance of separation between APs is chosen to be $d = 110m$, where neighboring APs do not overlap when both transmitting with lowest power while overlap otherwise.

For each particular experiment we report the average of 50 random instances for each algorithm or IP solution. For each experiment when running different algorithms the same 50 random instances are considered to make comparison fair. The next section reports results and insights from simulation.

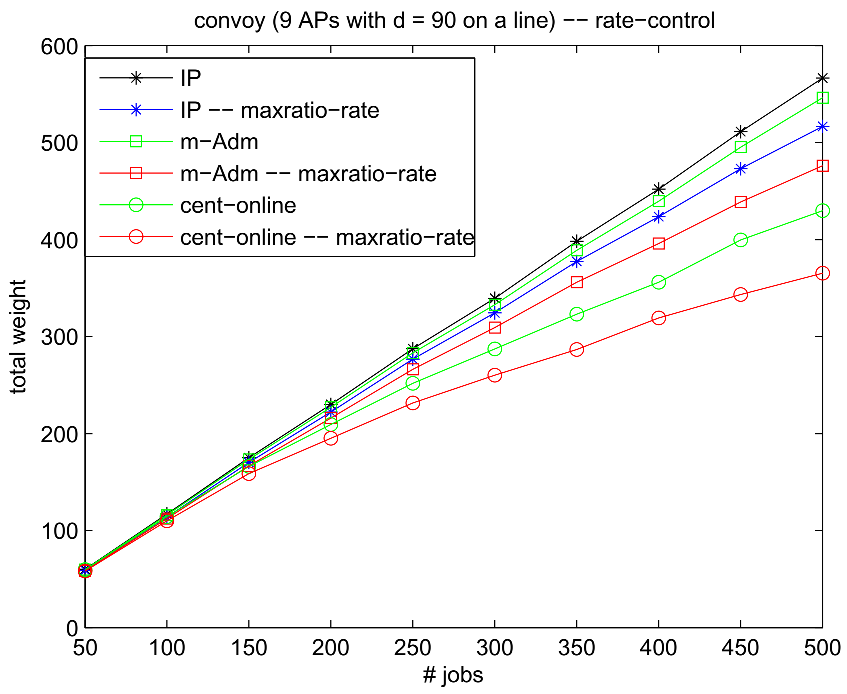


Figure 4.8: Rate-Controlled throughput for convoy

4.4.1 Results and Insights

The case of power control on one AP is depicted on Figure 4.4. The curves for IP with all power levels and for IP with only maximum power level coincide and are higher than the curve for IP with only minimum power level. This clearly shows that, since there are no other APs to interfere with, the optimal solution with power control is to always select the highest possible power for scheduling jobs. This is not true in a case of 3 APs deployed on a line. Since APs may interfere with one another the maximum power is not always optimal anymore. This can be seen from Figure 4.5 where the curve of IP with maximum power is lower than the curve of IP. Nonetheless, the curve of IP with maximum power is still higher than the curve of IP with minimum power. This shows that by increasing power we increase contact window size that may have sometimes more positive effect such as

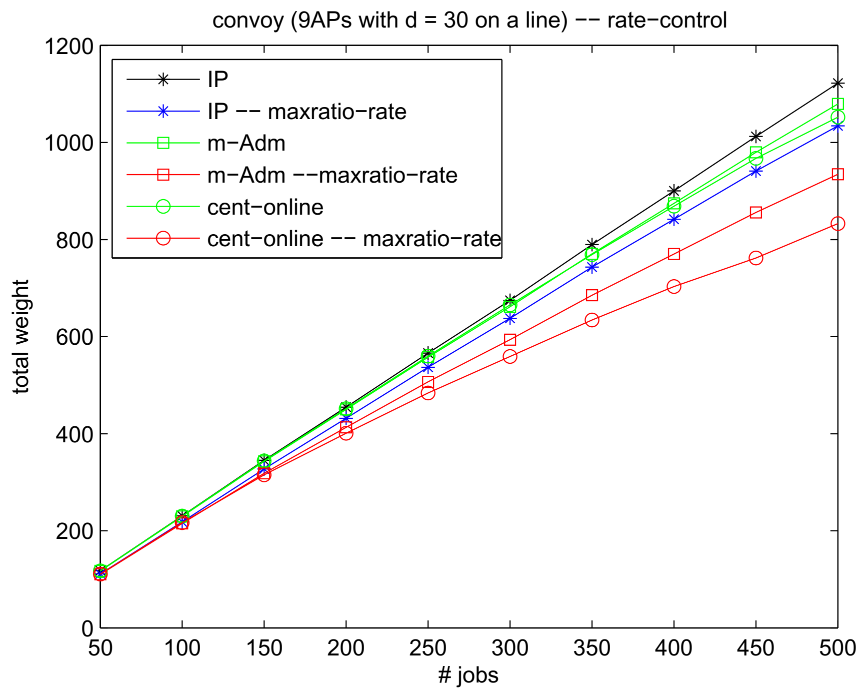


Figure 4.9: Rate-Controlled throughput for convoy

increased throughput than negative effect such as interference. Figure 4.6 shows runtimes in seconds for running IPs with all power levels or just maximum or minimum power level. One may observe that running IP with all possible power levels become prohibitively expensive really fast even with small number of jobs, whereas IPs with only single power level (i.e. max or min) the runtime does not increase so fast.

The case of rate control on one AP is depicted on Figure 4.7. From the figure we see that unlike with power control, selecting maximum rate is inefficient. This is due to the double effect that the rate has on scheduling, i.e., increasing rate not only shrinks the job size but also shrinks the contact window size. With lower rate, the contact window size increases, but in expense of increased job size. In fact employing only one fixed rate, whether min or max, gives very poor performance as is seen in the figure. Preselecting a rate where the

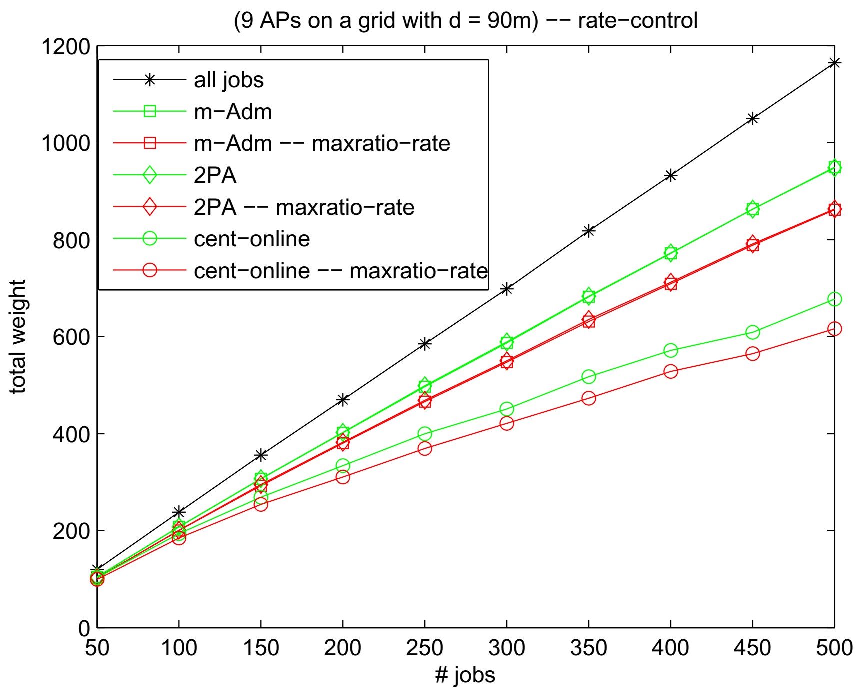


Figure 4.10: Rate-Controlled throughput for a grid

ratio of contact window size to job size is maximum gives better throughput than using any single fixed rate for all jobs; however, it is still sub-optimal to a case where all rate levels are considered.

Performance of algorithms for a convoy passing through 9 APs on a line separated by a distance of $d = 90m$ and $d = 30m$ is depicted on Figures 4.8 and 4.9 respectively. As seen from the Figure 4.8, the solution based on m-Admission algorithm gives a near optimal throughput. Even m-Admission with preselected rates has better performance than centralized-online algorithm that operates on all possible rates. However, when the separation between the APs decreases to $d = 30m$, as is seen from Figure 4.9, the centralized online algorithm gives a much better throughput, even outperforming the IP solution with preselected rates.

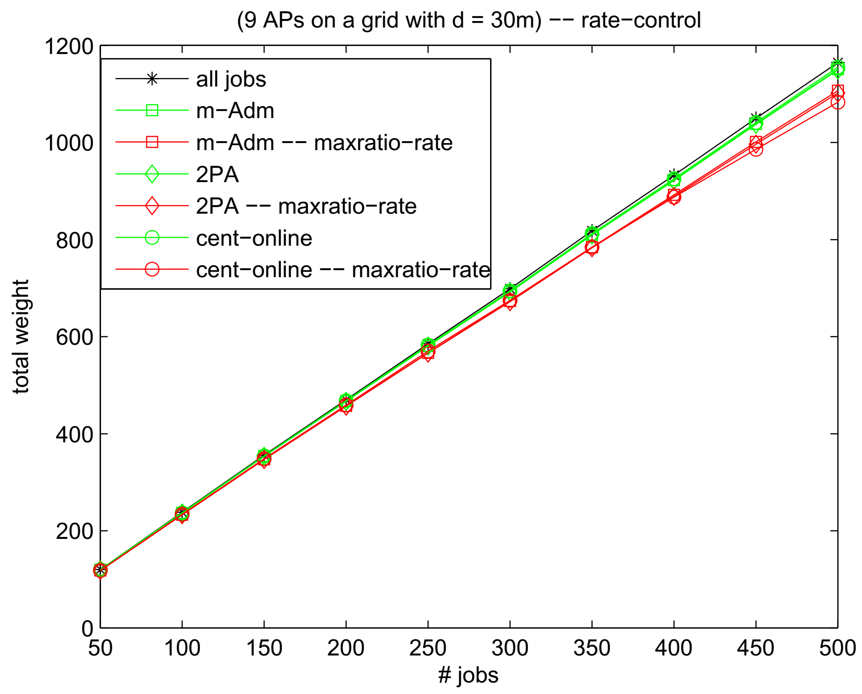


Figure 4.11: Rate-Controlled throughput for grid

The case of rate control with 9 APs on a grid is depicted on Figures 4.10 and 4.11. On Figure 4.10 we see that the lines for m-Admission and two-phase algorithms almost coincide. The centralized-online algorithms gives a slightly lower throughput. Performance of all three algorithms with preselected rates based on maximum ratio of contact window size to job size evidently give a lower throughput. However, we have observed that the runtime drops by an order of magnitude when best rates are preselected. The runtime is very crucial especially for the online setting when the schedule needs to be created online. When the APs are located closer on a grid all algorithms give near optimal solution as is seen from Figure 4.11.

Performance of 2PA-RPC algorithm for power-controlled throughput is shown on Figures 4.12 for one AP and 4.13 for multiple APs. In Figure 4.12 we have IP solutions for

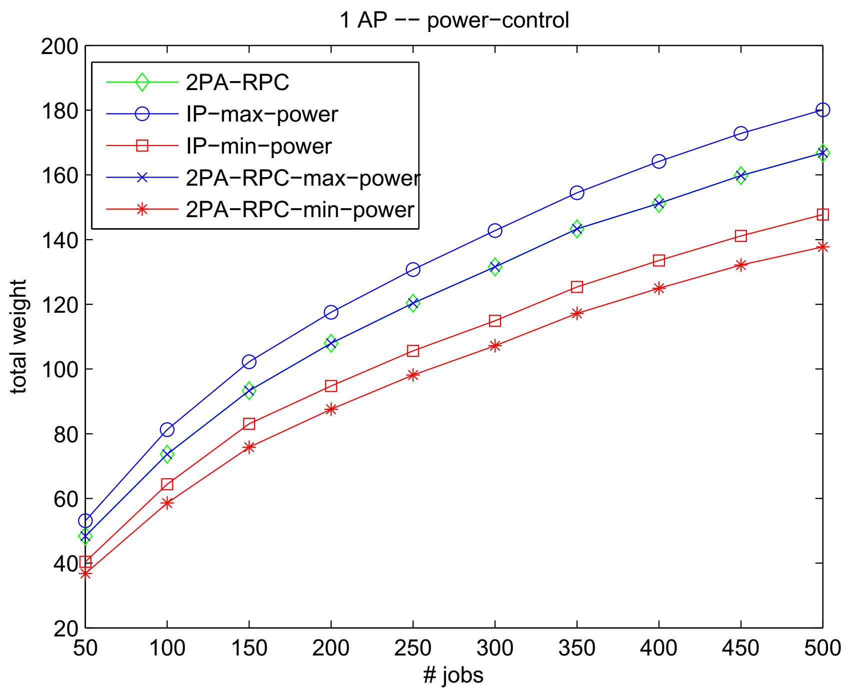


Figure 4.12: Power-Controlled throughput for 1AP

both using maximum transmission power and minimum transmission power. For one AP it is always optimal to use the maximum transmission power and, thus, the IP solution using maximum transmission power is optimal even when using all possible power levels. From the figure we see that the curve for 2PA-RPC algorithm is higher than IP-min-power curve, which means that 2PA-RPC algorithms gives better throughput than the best possible solution with only one lowest transmission power level. When running 2PA-RPC algorithm with only max power level it gives the same throughput as when running 2PA-RPC with all possible power levels. However, 2PA-RPC with min power level gives the lowest throughput, as expected. For the case of multiple APs using always the maximum power level is not always optimal but still better than using the minimum power level. In Figure 4.13 we see that our 2PA-RPC still gives better solution than just using lowest power level. We also

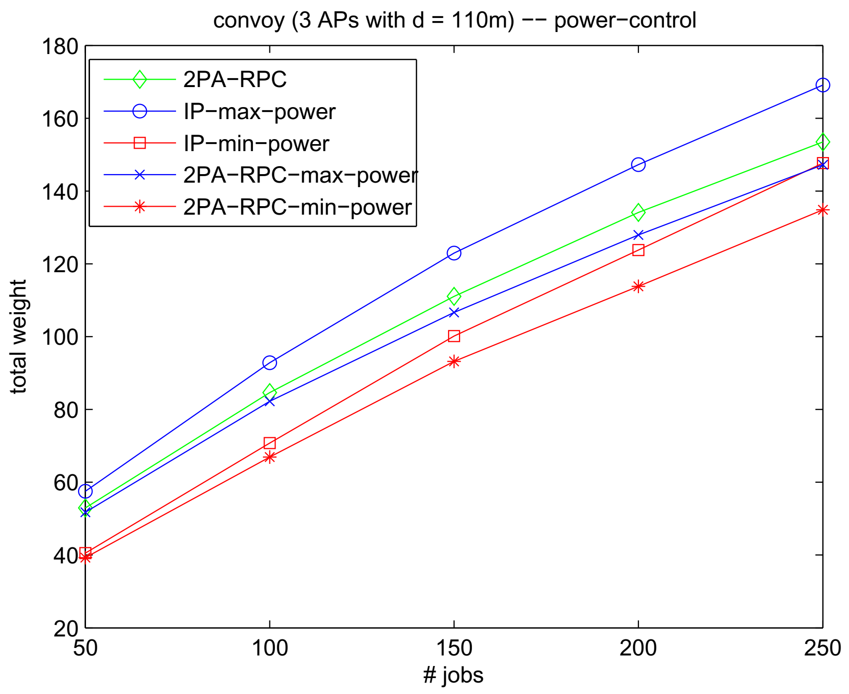


Figure 4.13: Power-Controlled throughput for convoy

compare the throughput of 2PA-RPC algorithm with the ones when running 2PA-RPC using only max or only min power levels and observe that 2PA-RPC gives better throughput than if using just max or min power levels.

4.5 Conclusion and Future Work

In this paper we have studied a variant of TMP problem with adaptive transmission power and rate control. We have formulated the problem for joint scheduling with either power control or rate control or both. We have adopted existing and proposed new algorithms with performance guarantees.

An interesting open problem is raised by our work. We have considered that when two APs transmit with such a power that creates an overlap between transmission circles, then APs interfere and can not transmit at the same time. However, it is a liberal assumption since the jobs do not have to be within an overlap region. If the two jobs receiving transmission from two APs are outside the overlap region then such overlap may still be considered. The solution to such a problem should take into account not just the duration of contact windows but also point to point location of jobs within a region. In such a case the performance guarantee of the 2PA-PC algorithm may be improved.

```
begin Enumeration stage:
    // For algorithm: $m$-Admission -- all rates;
    Enumerate all job instances $(j, k, \rho, s)$ ;
        OR
    // For algorithm: $m$-Admission -- max-ratio rate selection;
    For each $(j, k)$, pre-select $\rho^*$ such that $\rho^*_{j,k} = \arg_\rho \max \frac{d_{jk\rho} - r_{jk\rho}}{p_{jk\rho}}$;
    (i.e. Enumerate all job instances $(j, k, \rho^*_{j,k}, s)$);
end
Input: Enumerated job instances
Output: Feasible schedule of jobs on $m$ machines
begin Job Selection Stage:
    Let $S \longleftarrow \varnothing$ be admission schedule;
    for each machine $k$: do
        $I \longleftarrow$ the set of all job instances $(job, weight, beginning, ending)$;
        // Note: Consider the jobs that are not yet scheduled on previous machines;
        sort $I$ in order of non-decreasing ending (conflicts are resolved by
        considering higher weighted instances first, then if the weights are the same
        the order is according to bigger beginning);
        Let $A \longleftarrow \varnothing$ be schedule on machine $k$;
        while $I \neq \varnothing$ do
            let $J_j \in I$ be a job instance that terminates earliest;
            $I \longleftarrow I \setminus \{J_j\}$;
            if $J_j$ is not yet scheduled then
                let $C_j$ be the set of jobs in $A$ overlapping with $J_j$;
                let $W$ be the total weight of $C_j$;
                if $W = 0$ or $w_j > 2 \cdot W$ then
                    $A \longleftarrow A \cup \{J_j\} \backslash C_j$
                end
            end
        end
        Append all jobs in $A$ to $S$;
    end
    return $S$
end
```

Algorithm 7: m-Admission Algorithms

```
Let $A \longleftarrow \varnothing$ be global-admission schedule;
Let $I \longleftarrow$ the set of all job instances;
while $I \neq \varnothing$ do
    let $J_j \in I$ be a job instance that terminates earliest;
    $I \longleftarrow I \setminus \{J_j\}$;
    if $J_j \notin A$ then
        let $C_j \longleftarrow$ the set of jobs $\in A$ overlapping with $J_j$;
        let $W \longleftarrow$ the total weight of $C_j$;
        if $W = 0$ or $w_j > 2 \cdot W$ then
            $A \longleftarrow A \cup \{J_j\} \backslash C_j$
        end
    end
end
return $A$
```

Algorithm 8: Global-Admission Algorithm

```
for each moment $t$ and a new job $J_j$ arrives: do
    Fix all scheduled jobs $J_i$ with starting time $s \leq t$;
    Remove other jobs from the scheduled job list;
    Call Global-Admission with all unscheduled jobs;
end
```

Algorithm 9: Centralized-Online Algorithm

```
Let $tot(x.i)$ be the total *value* of job instances of $i$ on the stack;
Let $TOT(x)$ be the total *value* of instances of jobs *other than* $x.i$ on the stack, with
ending $> x.d$ and interfering with $(x.k, x.\pi, x.\rho)$;
for each job $i$ do
    $done[i] \longleftarrow false$;
end
begin Phase One: Evaluation
    $L \longleftarrow$ the set of all job instances $x = (i, w, d, e, k, \pi, \rho)$;
    sort $L$ in order of nondecreasing ending (ties are resolved by considering higher
    weighted instances first, then if the weights are the same the order is according to
    bigger beginning, and by machine arbitrarily; the approximation guarantee does
    not depend on this tie-breaking method);
    $S \longleftarrow$ an empty stack;
    for each $x$ from $L$ do
        $v \longleftarrow x.w - tot(x.i) - TOT(x)$;
        if $v > 0$ then
            push $((i, v, d, e, k, \pi, \rho), S)$;
        end
    end
end
begin Phase Two: Scheduling
    for each machine $k$ do
        $occupied[k] \longleftarrow t$;
    end
    while $S \neq \varnothing$ do
        $(i, v, d, e, k, \pi, \rho) \longleftarrow$ pop($S$);
        if $done[i] = false$ and $e \leq occupied[k]$ and $(i, v, d, e, k, \pi, \rho)$ does not
        interfere with any jobs already scheduled on other machines then
            add $(i, w, d, e, k, \pi, \rho)$ to solution;
            $done[i] \longleftarrow true$;
            $occupied[k] \longleftarrow d$;
        end
    end
end
```

Algorithm 10: Two Phase Algorithm – Rate-Power-Control (2PA-RPC)

Chapter 5

Conclusions

In this chapter, we first summarize the general research approach applied in this dissertation. We then highlight the contributions in this dissertation and conclude by proposing some future research directions.

5.1 General Approach

In this dissertation we have addressed several scheduling problems as applied to wireless sensor network environments. For each scheduling problem we study, we follow a similar general approach: modeling, analysis, theory, algorithm design, simulation and evaluation.

We first motivate the scheduling problem we study. We then develop a mathematical model of the real world problem with reasonable assumptions. Challenges in modeling lie in the complex nature of the wireless network systems and thus simplifications are inevitable. By carefully making assumptions we may obtain tractable abstract problem without losing sufficient factors that lead to practicable solutions.

Linear, Non-Linear, or Integer Programs are often used to formulate the optimization problem precisely. In special cases we can solve Integer Programs for the problem efficiently and fast. In others we relax integral constraints and solve LP-relaxation. Linear or Integer Programs help with the analysis and most of the time give insights into the solutions as well as provide either an optimal or a bound to an optimal solution for the purpose of evaluation.

A typical objective of the scheduling problems we study is to maximize some profit (i.e. throughput) or minimize some loss (i.e. data obsoleteness, limited observation, battery charge) subject to constraints. The complexity of constraints of a real wireless sensor network system makes optimization difficult. In most general cases we prove NP-hardness of the scheduling problems and aim for efficient and fast approximate solutions. Some special cases though admit polynomial time algorithms. Algorithms design for general settings extends theoretical results. Proofs of performance guarantees have contributed to the depth of the work.

Sound theoretical performance of algorithms needs to be verified in realistic settings. The challenges lie in creation of realistic network environment. By carefully designing evaluation logic and controlling variable factors we have implemented simulation environment to verify and evaluate algorithms performance as well as compare performance of algorithms among each other.

5.2 Contributions

In this dissertation we solve problems of allocating sensor or network resources in several sensor network scenarios. These problems seek to optimize the utilization of sensor and network resources and provide guaranteed performance with respect to complex realistic constraints using scheduling methods. In most general settings these problems are difficult to solve (NP-hard in most cases). Our goal is, therefore, to provide efficient heuristic or approximate solutions.

In Chapter 2 we have studied a specialized monitoring application in wireless sensor networks in which sensors execute dynamic missions and require new mission updates sent to them from a command center via a shared time-division broadcast channel [7, 8]. Since conservation of energy supplied by sensor batteries is of the essence, sensors are normally put to sleep when not actively listening. Schedule is required in order to specify when sensors should listen for updates and when they should sleep. However, rebooting can be so costly as to consume more energy than is saved during the time spent in sleep mode. Thus, we have formulated a new model that incorporates data-related and sensor-related costs recognizing the cost of sensor rebooting. We have derived and proved an optimal solution to the scheduling of one sensor. Based on the optimal solution to one sensor we have formulated and solved a mathematical program that serves as a lower bound on the cost of scheduling configurations with many sensors. Since our scheduling problem remains NP-hard even in the restricted case where there is no reboot cost, we have introduced several heuristic algorithms for scheduling multiple sensors, compared their performances, and demonstrated various trade offs among the cost factors.

In Chapter 3 we have studied a single sensor scheduling problem (SSSP), motivated by monitoring applications in wireless sensor networks where the goal is to maintain up-to-date readings of all the observed sites [6]. We have formulated a model for the sensor scheduling problem. Prior research has neglected the pan and tilt delay time to refocus sensor from one site to the next. We have incorporated sensor refocus time in our model. We have proved the problem to be NP-hard using Cook reduction from Maximum Independent Set. We have also proved and shown that every SSSP problem with symmetric delay constraints admits an optimal solution that is periodic. We have studied special cases in which refocusing is proportional to some Euclidian metric. We have provided and proved exact solutions for the scheduling problems in the setting of two sites. We have formulated and solved a mathematical program that serves as a lower bound on the cost of scheduling configurations with many sites. We have solved small instances of the SSSP exactly with IP. Since in most general settings, however, the SSSP remains NP-hard we have introduced new greedy based heuristics. Some of the algorithms use the solution obtained to the lower bound while others use cost parameters directly. Chained-IP algorithm, which is shown to be the best, uses IP-formulation directly to implement a greedy solution with many steps look-ahead. We have evaluated these algorithms in the synthetic simulation study and compared their performance with the lower bound cost as well as compared performance among all the algorithms.

In chapter 4 we have studied variations of a problem in which data must be delivered to mobile clients en-route, as they travel towards their destinations [9, 25, 26]. We have casted this scenario as a parallel-machine scheduling problem with the little-studied property that release times and deadlines are machine-dependant. We have addressed an open issue of how to adaptively control the transmission rate and power to optimize the matching/scheduling between end users and machines to maximize the throughput. The problem is a generalization of an already NP-hard parallel-machine scheduling problem in which jobs' release times and deadlines are different. We have defined variations of models where either transmission rate is controlled, transmission power is controlled, or both transmission rate and power are controlled. We have solved small instances of the problem exactly using IP. We have studied both online and offline settings. For the general instances of the problem we have introduced new algorithms as well as adapted existing algorithms with guaranteed approximation ratios. For the general setting of the problem with rate and power control, we have provided a combinatorial algorithm and proved its approximation guarantee. We have evaluated these algorithms on a variety of problem instance types using synthetic data for several geographic scenarios. We have shown that our algorithms produce schedules achieving near-optimal throughput.

5.3 Future Directions

In this dissertation we have shown how the broadest of the wireless sensor network problems, utilizing limited networking and sensor resources optimally under constraints, can be successively narrowed to obtain a quite specific, natural problem, which is of interest in modern emergency and rescue settings. We studied several resulting scheduling problems to which we seek efficient algorithms. The problems studied in this dissertation lead to some interesting open research directions.

In one direction we can address some of the open problems in the current models of the scheduling problems that we study in this dissertation. In Chapter 2, even though we have derived a lower bound for the minimum cost of the schedule for the broadcast scheduling of info-pages to sensors problem, it remains an open problem to employ this lower bound to approximate the minimum cost within a ratio that could be quantified analytically or numerically. Designing a greedy solution that is a 2-approximation to the fractional solution of a lower bound would be interesting. In Chapter 3, we have proved that a single sensor scheduling problem with refocusing constraints is an NP-hard problem. However, under assumption that the refocusing delays are all zero, it remains an open problem if the single sensor scheduling problem is NP-hard in the restricted setting. It would be interesting to prove or disprove NP-hardness in the restricted setting of the problem. In Chapter 4, in data dissemination problem for the setting where only rates can be controlled we have introduced m-Admission and Two-Phase based algorithms that preselect the best existing rate for each job-AP pair. The algorithms will run faster on expense of losing the approximation guarantees. It would be an interesting problem to perform approximation analysis of these algorithms that preselect the best rate and derive the approximation ratios.

In another direction, models can be generalized to incorporate other variables and constraints. We can then study new variations of the scheduling problems. There are several ways to generalize broadcast scheduling of updates to sensors: to allow unequal-sized info-pages, use multiple broadcast channels, impose precedence constraints for task updates, and allow preemption and/or migration of broadcast task updates. Each of these extensions would be interesting both theoretically as well as in terms of more realistic applications. For the single sensor scheduling, refocusing a sensor may consume energy due to mechanical movement of the camera. Energy cost can be quantified and incorporated into our optimization problem. Furthermore, a single sensor scheduling can be extended to multiple-sensor scheduling. One way to obtain a multiple-sensor scheduling instance is to optimally partition the collection of sites into disjoint sets and let each individual sensor be responsible for its own set of sites. The challenge is in solving the partition of the sites problem either online or offline. Obtaining an efficient solution requires techniques such as data clustering and machine learning. For data dissemination scheduling we have assumed that when transmission circles of two APs overlap, then the APs interfere and, thus, can not transmit at the same time to two different users. However, it is a liberal assumption since the users do not have to be within the overlap region. If the users are not within the overlap region, the two APs should still be able to transmit without a problem. Formulating a point to point location of users within a region and incorporating them into the IP is an open problem.

Yet a third interesting direction is to study users' habits, such as data preferences and user mobility patterns, to better address our scheduling problems. For broadcast scheduling we have assumed that the system knows about task popularities. For sensor scheduling we have assumed the knowledge of the importance of the sites. For the dissemination scheduling we have assumed the knowledge of data preferences and user mobility. In practice, however, obtaining such information requires data mining and machine learning techniques. A study of learning users' habits to predict their needs and movements may lead to more accurate and efficient schedules.

Bibliography

[1] *Bluetooth technical specifications, version 2.0. available from http://www.bluetooth.com/*, 2004.

[2] *Smartrf cc2420 data sheet, chipcon*, 2004.

[3] *Momentus 5400.3 2.5-inch storage for mainstream notebook pcs, seagate*, 2005.

[4] Swarup Acharya, Rafael Alonso, Michael J. Franklin, and Stanley B. Zdonik, *Broadcast disks: Data management for asymmetric communications environments*, SIGMOD Conference, 1995, pp. 199–210.

[5] Swarup Acharya, Michael Franklin, and Stanley Zdonik, *Dissemination-based data delivery using broadcast disks*, IEEE Personal Communications **2** (1995), no. 6, 50–60.

[6] Yosef Alayev, Amotz Bar-Noy, Matthew P. Johnson, Lance M. Kaplan, and Thomas F. La Porta, *You can't get there from here: Sensor scheduling with refocusing delays*, MASS, 2010, pp. 462–471.

[7] Yosef Alayev, Amotz Bar-Noy, and Thomas F. La Porta, *Broadcasting info-pages to sensors: Efficiency vs. energy conservation*, SECON, 2008, pp. 368–376.

[8] ———, *Broadcasting info-pages to sensors: efficiency versus energy conservation*, Wireless Networks **17** (2011), no. 6, 1529–1542.

[9] Yosef Alayev, Fangfei Chen, Yun Hou, Matthew P. Johnson, Amotz Bar-Noy, Tom La Porta, and Kin K. Leung, *Throughput maximization in mobile wsn scheduling with power control and rate selection*, DCOSS, 2012, pp. 33–40.

[10] Ali Allahverdi, C.T. Ng, T.C.E. Cheng, and Mikhail Y. Kovalyov, *A survey of scheduling problems with setup times or costs*, European Journal of Operational Research **187** (2008), no. 3, 985–1032.

[11] Mostafa Hamed Ammar and J. W. Wong, *The design of teletext broadcast cycles*, Performance Evaluation **5** (1985), no. 4, 235–242.

[12] ______, *On the optimality of cyclic transmission in teletext systems*, IEEE Transactions on Communications **35** (1987), no. 1, 68–73.

[13] Shoshana Anily, Celia A. Glass, and Refael Hassin, *The scheduling of maintenance service*, Discrete Applied Mathematics **82** (1998), no. 1-3, 27–42.

[14] Nilanjan Banerjee, Jacob Sorber, Mark D. Corner, Sami Rollins, and Deepak Ganesan, *Triage: Balancing energy consumption and quality of service in a microserver*, in ACM/USENIX Conference on Mobile Systems, Applications, and Services (MobiSys, 2007, pp. 152–164.

[15] Philippe Baptiste, *Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times*, Journal of Scheduling **2** (1999), no. 6, 245–252.

[16] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber, *A unified approach to approximating resource allocation and scheduling*, Journal of the ACM **48** (2001), no. 5.

[17] Amotz Bar-Noy, Randeep Bhatia, Joseph (Seffi) Naor, and Baruch Schieber, *Minimizing service and operation costs of periodic scheduling*, Math. Oper. Res. **27** (2002), 518–544.

[18] Amotz Bar-Noy, Sudipto Guha, Joseph (Seffi) Naor, and Baruch Schieber, *Approximating the throughput of multiple machines in real-time scheduling*, SIAM Journal of Computing **31** (2001), no. 2.

[19] Amotz Bar-Noy and Richard E. Ladner, *Windows scheduling problems for broadcast systems*, SIAM Journal on Computing **32** (2003), 433–442.

[20] Piotr Berman and Bhaskar Dasgupta, *Improvements in throughput maximization for real-time scheduling*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, 2000, 1999, pp. 680–687.

[21] Piotr Berman and Bhaskar DasGupta, *Multi-phase algorithms for throughput maximization for real-time scheduling*, J. Comb. Optim. **4** (2000), no. 3, 307–323.

[22] Nikita Boyko, Timofey Turko, Vladimir Boginski, David E. Jeffcoat, Stanislav Uryasev, Grigoriy Zrazhevsky, and Panos M. Pardalos, *Robust multi-sensor scheduling for multi-site surveillance*, Journal of Combinatorial Optimization **ISSN 1382-6905, DOI 10.1007/s10878-009-9271-4** (2009).

[23] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva, *A performance comparison of multi-hop wireless ad hoc network routing protocols*, Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (New York, NY, USA), MobiCom ’98, ACM, 1998, pp. 85–97.

[24] Peter Brucker, *Scheduling algorithms*, Springer, 2007.

[25] Fangfei Chen, Matthew P. Johnson, Yosef Alayev, Amotz Bar-Noy, and Thomas F. La Porta, *Who, when, where: Timeslot assignment to mobile clients*, IEEE Transactions on Mobile Computing **11** (2012), 73–85.

[26] Fangfei Chen, Matthew P. Johnson, Yosef Alayev, Amotz Bar-Noy, and Tom La Porta, *Who, when, where: Timeslot assignment to mobile clients.*, MASS, IEEE, 2009, pp. 90–99.

[27] Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani, *Approximation algorithms for the job interval selection problem and related scheduling problems*, Math. Oper. Res. **31** (2006), 730–738.

[28] Cash J. Costello, Christopher P. Diehl, Amit Banerjee, and Hesky Fisher, *Scheduling an active camera to observe people*, VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks (New York, NY, USA), ACM, 2004, pp. 39–45.

[29] Amos Fiat, Yuval Rabani, and Yiftach Ravid, *Competitive k-server algorithms*, J. Comput. Syst. Sci. **48** (1994), no. 3, 410–428.

[30] Gabor Galambos and Gerhard J. Woeginger, *Online bin packing a restricted survey*, Mathematical Methods of Operations Research **42** (1995), no. 1, 25–45.

[31] Michael R. Garey and David S. Johnson, *Computers and intractability: A guide to the theory of np-colmpleteness*, Freeman, 1979.

[32] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, *Optimization and approximation in deterministic sequencing and scheduling: A survey*, Annals of Discrete Mathematics **5** (1979), 287–326.

[33] Nicholas G. Hall and Michael J. Magazine, *Maximizing the value of a space mission*, European journal of operational research **78** (1994), no. 2, 224–241.

[34] Claire Hanen and Alix Munier, *Cyclic scheduling on parallel processors: an overview*, Scheduling Theory and its Applications (1994), 193–226.

[35] J. A. Hoogeveen, J. K. Lenstra, and S. L. van de Velde, *Sequencing and scheduling: an annotated bibliography*, Memorandum COSOR 97-02 (1997).

[36] Fethi Jarray, *Complexity results for wireless sensor network scheduling.*, Wireless Sensor Network **2** (2010), no. 5, 343–346.

[37] Konstantin Kalinchenko, Alexander Veremyev, Vladimir Boginski, David E Jeffcoat, and Stan Uryasev, *Robust connectivity issues in dynamic sensor networks for area surveillance under uncertainty*, Structural and Multidisciplinary Optimization **7** (2011), no. 2, 235–248.

[38] Sanjeev Khanna and Shiyu Zhou, *On indexed data broadcast*, Proceedings of the thirtieth annual ACM symposium on Theory of computing (New York, NY, USA), STOC '98, ACM, 1998, pp. 463–472.

[39] Antoon W.J. Kolen, Jan Karel Lenstra, Christos H. Papadimitriou, and Frits C.R. Spieksma, *Interval scheduling: A survey*, Naval research logistics **54** (2007), no. 5, 530–543.

[40] Youngho Lee and Hanif D. Sherali, *Unrelated machine scheduling with time-window and machine downtime constraints: An application to a naval battle-group problem*, Annals of Operations Research **50** (1994), 339–365.

[41] Edward Yu-Hsien Lin, *A bibliographical survey on some well-known non-standard knapsack problems*, INFOR **36** (1998), no. 4, 274–317.

[42] Hang Liu and Magda El Zarki, *Adaptive source rate control for real-time wireless video transmission*, Mobile Networks and Applications, 1998, pp. 49–60.

[43] Jeongyeup Paek and Ramesh Govindan, *Rcrt: Rate-controlled reliable transport protocol for wireless sensor networks*, ACM Trans. Sen. Netw. **7** (2010), no. 3, 20:1–20:45.

[44] Wei pang Chin and Simeon C. Ntafos, *Optimum watchman routes*, Symposium on Computational Geometry, 1986, pp. 24–33.

[45] Cong Peng, Fan Yang, Qian Zhang, Dapeng Wu, Ming Zhao, and Yan Yao, *Impact of power and rate selection on the throughput of ad hoc networks*, IEEE International Conference on Communications. ICC 06., vol.9, 2006, pp. 3898–3902.

[46] Jan M. Rabaey, Josie Ammer, Tufan Karalar, Suetfei Li, Brian Otis, Mike Sheets, and Tim Tuan, *Picoradios for wireless sensor networks: The next challenge in ultra-low-power design*, Proceedings of the International Solid-State Circuits Conference, February 2002.

[47] Jan M. Rabaey, M. Josie Ammer, Julio L. da Silva Jr., Danny Patel, and Shad Roundy, *Picoradio supports ad hoc ultra-low power wireless networking*, IEEE Computer Magazine **33** (2000), no. 7, 42–48.

[48] Vijay Raghunathan, Curt Schurgers, Sung Park, Mani Srivastava, and Barclay Shaw, *Energy-aware wireless microsensor networks*, IEEE Signal Processing Magazine, 2002, pp. 40–50.

[49] Sumit Rangwala, Ramakrishna Gummadi, Ramesh Govindan, and Konstantinos Psounis, *Interference-aware fair rate control in wireless sensor networks*, In Proceedings of the ACM SIGCOMM, 2006, pp. 63–74.

[50] Jonathan M. Reason and Jan M. Rabaey, *A study of energy consumption and reliability in a multi-hop sensor network*, ACM Mobile Computing and Communications Review **8** (2004), no. 1, 84–97.

[51] Harvey M. Salkin and Cornelis A. De Cluyver, *The knapsack problem: A survey*, Naval Research Logistics Quarterly **22** (1975), no. 1, 127–144.

[52] Claude Shannon, Noshirwan Petigara, and Satwiksai Seshasai, *A mathematical theory of communication*, Communication, Bell System Technical Journal **27** (1948), 379–423 and 623–656.

[53] Yi Shi, Thomas Y. Hou, Jia Liu, and Sastry Kompella, *How to correctly use the protocol interference model for multi-hop wireless networks*, Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing (New York, NY, USA), MobiHoc '09, ACM, 2009, pp. 239–248.

[54] Abraham Silberschatz, Peter Galvin, and Greg Gagne, *Operating system concepts, sixth edition*, John Wiley, 2002.

[55] Barbara B. Simons and Manfred K. Warmuth, *A fast algorithm for multiprocessor scheduling of unit-length jobs*, SIAM Journal of Computing **18** (1989), no. 4.

[56] Yang Song, Chi Zhang, and Yuguang Fang, *Throughput maximization in multi-channel wireless mesh access networks*, IEEE International Conference on Network Protocols (ICNP) (2007), 11 – 20.

[57] Stanislav Uryasev and Panos Pardalos, *Detecting and jamming dynamic communication networks in anti-access environments*, Tech. report, DTIC Document, 2011.

[58] Andrew Wang, SeongHwan Cho, Charles Sodini, and Anantha Chandrakasan, *Energy efficient modulation and mac for asymmetric rf microsensor systems*, Proceedings of the 2001 international symposium on Low power electronics and design (New York, NY, USA), ISLPED '01, ACM, 2001, pp. 106–111.

[59] John-Patrick Wowra, *Approaches to reduce energy consumption of wlan devices*, 2004.

[60] Yiliang Xu and Dezhen Song, *Systems and algorithms for autonomous and scalable crowd surveillance using robotic ptz cameras assisted by a wide-angle camera*, Auton. Robots **29** (2010), no. 1, 53–66.

[61] David K. Y. Yau and Simon S. Lam, *Adaptive rate-controlled scheduling for multimedia applications*, IEEE/ACM Transactions on Networking, 1996.

[62] Mesut Yavuz and David E. Jeffcoat, *An analysis and solution of the sensor scheduling problem*, pp. 167–177, Springer Berlin / Heidelberg, October 2007.

[63] ______, *Single sensor scheduling for multi-site surveillance*, Tech. report, Air Force Research Laboratory, November 2007.