An Analysis and Solution of the Sensor Scheduling Problem

Mesut Yavuz[1] and David Jeffcoat[2]

[1] Research and Engineering Education Facility, University of Florida, Shalimar, FL
yavuz@reef.ufl.edu
[2] Air Force Research Laboratory, Munitions Directorate, Eglin AFB, FL
david.jeffcoat@eglin.af.mil

Abstract. This chapter addresses the scheduling problem of a sensor that constantly collects information from multiple sites. In the existing literature, the problem is solved by probabilistic approaches, potentially generating schedules in which a site is not visited for a long time. To overcome this deficiency, this chapter presents a deterministic approach formulated as an integer linear program. Upon showing that the problem is NP-Hard, the chapter develops valid lower and upper bounds and proposes two constructive heuristic methods. Tested via an extensive computational study, the heuristic methods are proven efficient and effective in solving the problem.

1 Introduction

This chapter is concerned with scheduling a single sensor to maintain an estimate of a dynamic physical attribute (e.g., position) of multiple targets. The research builds on previous work by Tiwari et al. [9], Yerrick et al. [11] and Yerrick et al. [12]. Tiwari et al. [9] present a feasibility criterion for a single sensor to maintain a bounded estimate of an attribute at multiple locations. Yerrick et al. [11] demonstrate by simulation the feasibility criterion presented in [9] and develop a heuristic to find a good sensor motion model given the dynamics of the system under observation. Yerrick et al. [12] provide an optimal sensor coverage solution for two sensor motion models given a model of the observed system's dynamics. All three papers consider probabilistic strategies for the motion of the single sensor among the sites. A similar model in the literature is known as the traveling inspector model [4, 5]. In this chapter, we focus on deterministic methods to schedule the sensor's motion. A deterministic approach overcomes one disadvantage of probabilistic motion: with any random motion strategy, there is nonzero probability that a particular site will not be visited at all in any finite time horizon.

Figure 1 provides an illustration for a three-site scenario. At the time instant pictured, the sensor is focused on site three. In its current position, the sensor can observe the characteristics of site three, but cannot observe sites one or two. In the next discrete time step, we assume that the sensor can move (or refocus) from site three to either of the other two sites, or can maintain its

M.J. Hirsch et al. (Eds.): Adv. in Cooper. Ctrl. & Optimization, LNCIS 369, pp. 167–177, 2007.
springerlink.com

current position. At each time step, since the sensor focuses on exactly one of the sites, the sensor's processor can update its information for only one site and must estimate the rest. Therefore, information loss at a site increases with the number of time steps that the site has not been visited. Sites may have different rates of change, and, hence, the criticality of information loss may vary among the sites. A successful sequence is one that balances the visit frequencies of the sites to minimize overall information loss. The goal of this chapter is to develop methods to construct such sequences.

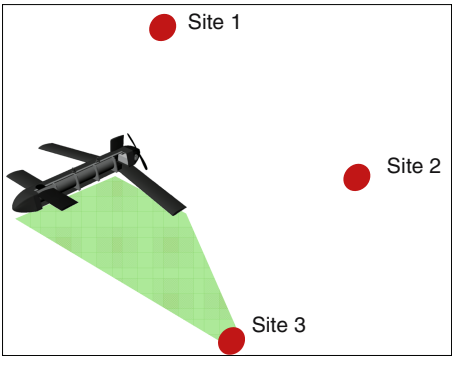

Fig. 1. Three site example

The remainder of this chapter is organized as follows. In Section 2 we formulate the problem as an integer linear programming problem and in Section 3 we analyze its properties. In Section 4 we develop a lower and an upper bound on the objective function of our model. In Section 5 we propose two heuristic solution procedures and in Section 6 we evaluate their performance. In Section 7, we conclude by summarizing our contribution and discussing possible future research directions.

2 A Mathematical Model

Let $x_{i,t}$ be the binary decision variable denoting whether the sensor is scheduled to visit site i at time t, and $y_{i,t}$ denote the last time site i was visited as of the end of time t. Note that $y_{i,t} = t$ happens only in time intervals in which the sensor visits site i. When the sensor is focused on site i, it updates the status of the site. In other words, we have perfect information of the site in that time step. Since the sensor cannot focus on more than one site at the same time, focusing on one site means losing information about the current states of the other sites. The extent of the information loss depends on the activity rate of a site. We can afford to ignore less active sites for a large number of time steps, whereas more active sites must be visited frequently. In this chapter, we assume that there is no cost for movement or observation; our whole concern is the cost of lost information.

We associate with each site i a fixed cost a_i and a variable cost b_i of information loss. More specifically, a fixed penalty of not visiting a certain site is incurred for each time step in which the sensor is away from the site. In addition, a variable cost is incurred for each time unit that has passed since the sensor's last visit to that site, providing ever-increasing motivation for the sensor to return to a neglected site. (The cost parameters implicitly model the activity level at a site or the importance of a site.) The objective function in our model minimizes the maximum penalty incurred for a sensor schedule defined over a finite time horizon. If we are given a planning horizon consisting of T periods, then the following integer linear program can be formulated.

$$\begin{aligned}
&\text{Minimize} && C && && (1)\\
&\text{Subject to} && && && \\
&C + a_i x_{i,t} + b_i y_{i,t} \geq a_i + b_i t, && && \forall i = 1,..,n; \forall t = 1,..,T && (2)\\
&\sum_{i=1}^{n} x_{i,t} = 1, && && \forall t = 1,..,T && (3)\\
&y_{i,t} - y_{i,t-1} \leq t x_{i,t}, && && \forall i = 1,..,n; \forall t = 1,..,T && (4)\\
&y_{i,t} \leq t, && && \forall i = 1,..,n; \forall t = 1,..,T && (5)\\
&y_{i,0} = 0, && && \forall i = 1,..,n && (6)\\
&C > 0, && && && (7)\\
&x_{i,t} \in \{0,1\}, && && \forall i = 1,..,n; \forall t = 1,..,T && (8)\\
&y_{i,t} \in \{0\} \cup \mathbb{Z}^+, && && \forall i = 1,..,n; \forall t = 1,..,T && (9)
\end{aligned}$$

The model is built as a fully linear model, that is, the objective function and constraints are all linear functions of the decision variables. Note that defining C as a variable and defining it in a constraint is critical for the linearity of the formulation. The objective function of the model (1) simply aims to minimize the maximum cost defined by the first constraint (2). More specifically, $C \geq a_i(1 - x_{i,t}) + b_i(t - y_{i,t})$, for all i and t. Constraint (3) assures that the sensor visits exactly one site in each stage. Constraints (4) and (5) together assure that $y_{i,t}$ is updated only when the sensor is on site i, and remains constant at other times. Constraint (6) initializes variable y. Finally, constraints (7-9) define the decision variables C, x and y as nonnegative, binary and nonnegative-integer variables, respectively.

3 Structural Properties of the Problem

The optimization model of the previous section is built upon a given sequence length, T. However, in practice, we may not be given such a length but asked to find infinitely long sequences. This property, regardless of the computational complexity of the formulated integer programming model, makes the problem a challenging one. This property motivates us to study the problem from a different perspective, that is, periodic scheduling. If an infinite sequence can be

constructed such that the objective function value is C, every site i must be visited by a period p_i where $a_i + (p_i - 1)b_i \leq C$ and $a_i + p_i b_i > C$. Various periodic scheduling problems arise in the context of computer and telecommunications systems, and have received significant academic interest. Consider a satellite with finite memory capacity that needs to download its memory periodically. If we have multiple satellites each serviced by a single download facility, then construction of a download sequence would constitute a periodic scheduling problem. The first result we use from the periodic scheduling literature proves the computational complexity of the sensor scheduling problem as follows.

Theorem 1. *The sensor scheduling problem is NP-hard.*

Proof. Bar-Noy et al. [1] show that the periodic scheduling problem is NP-hard, with a reduction from the graph coloring problem. Here, we reduce the periodic scheduling problem to our problem. An instance of the periodic scheduling problem is given as follows.

Given m machines and service intervals $p_1, p_2, .., p_m$ such that $\rho = \sum_{i=1}^{m} 1/p_i \leq 1$, does there exist an infinite maintenance service schedule of these machines in which consecutive maintenance times for machine i are exactly p_i time-slots apart and no more than one machine is serviced in a single time-slot?

For a given instance of the periodic scheduling problem, we first create m sites with $a_i = 0$ and $b_i = C/p_i$, where C is an arbitrarily selected constant. Next, we find the smallest positive integers c and d such that $c/d = 1 - \sum_{i=1}^{m} 1/p_i$ (if $\sum_{i=1}^{m} 1/p_i = 1$, then we assign $c = d = 0$). Then we create c additional sites each with $a_i = 0$ and $b_i = C/d$. Note that $d = 0$ is only possible when $c = 0$, in which case no additional sites are created. If we can find a solution to this problem with $n = m + c$ sites such that maximum cost is at most C, then in that solution the first m sites will be visited exactly every p_i time-slots, since the density (ρ) is now 1. □

Theorem 2. *There exists an optimal solution in which no site is visited in two consecutive stages.*

Proof. It is clear that when $n > 1$, the minimum cost for site $i = 1, .., n$ will be at least $a_i + b_i$, since at least one of the other sites must be visited between two consecutive visits to site i. Therefore, at any stage in the sequence, staying at the same site results in a zero cost for site i, whereas it increases the variable cost for all other sites $i' \neq i$ by $b_{i'}$. Hence, staying at the same site can only increase the maximum cost with respect to the other sites and it can never decrease the maximum cost factor at that site. Using this property, any optimal solution to the problem can be converted to another optimal solution in which no site is visited in two consecutive stages. □

Corollary 1. *Instances with two sites ($n = 2$) are trivial.*

Proof. This result directly follows from Theorem 2: if an optimal solution exists such that the sensor never stays at the same site in two consecutive stages and there are only two sites, then in each stage there is exactly one site that the sensor can focus on. □

Corollary 2. $\max_i(a_i + b_i)$ *is a lower bound for* C.

Proof. From Theorem 2 we know that an optimal solution to the sensor's schedule can be found by constantly moving between the sites. Therefore, there will be at least one time-slot in which a site (i) is not visited, thus the cost incurred at site i will be at least $a_i + b_i$. Since the objective function C is greater than or equal to those cost factors, it must be at least as large as the largest of them, which completes the proof. □

4 Lower and Upper Bounds on the Objective Function

Corollary 2 provides a loose lower bound on C. Before obtaining a tight lower bound, we first elaborate our discussion on periodic scheduling. A special version of the periodic scheduling class of problems is known as *pinwheel scheduling*, see [2, 3] for further reading. In the pinwheel scheduling problem, a number (n) of tasks each with a possibly distinct period (p_i) are aimed to be scheduled such that two consecutive executions of task i are not separated by more than p_i time steps. The sensor scheduling problem reduces to the pinwheel scheduling problem for a given C, and, hence, is a general case thereof. An instance of the pinwheel scheduling problem is characterized by its density $\rho = \sum_{i=1}^{n} 1/p_i$. It is well known that instances with $\rho > 1$ cannot be scheduled. Instances with $\rho \leq 1$ may or may not be scheduled. A widely believed conjecture is that all instances with $\rho \leq 5/6$ are schedulable. However, no one to date has been able to prove or disprove this conjecture.

We use the properties of the pinwheel scheduling problem to develop a tight lower bound and conjecture an upper bound. Both bounds are obtained using the search procedure, i.e., Algorithm `Search_on_`$C(n,\mathbf{a},\mathbf{b},\rho^U)$, depicted in Figure 2. The algorithm first uses Corollary 2 to find the minimum C value that is possible, and then performs an increasing search on C until a pinwheel instance with a density less than or equal to the designated threshold is obtained. In a basic setting, if $\mathbf{a}$ and $\mathbf{b}$ are integer vectors, the search can be performed by increasing C by one. Our algorithm performs the search intelligently in that it calculates the smallest candidate for the increased C value that will change at least one p_i value. Therefore, it is guaranteed that every time C is increased, a pinwheel instance with a lower density is obtained. We also denote the optimal solution of the sensor scheduling problem by C^*.

Theorem 3. $C^L =$ *Search_on_*$C\ (n,\mathbf{a},\mathbf{b},1)$ *is a lower bound for* C^*.

Proof. The proof is based upon the following two simple observations: ρ is non-increasing in C and there is no feasible schedule with $\rho > 1$. Thus, terminating the search when $\rho \leq 1$ assures that the minimum C value that may be schedulable is returned. □

Proposition 1. $C^C =$ *Search_on_*$C(n,\mathbf{a},\mathbf{b},5/6)$ *is always schedulable, and, hence, is an upper bound for* C^*.

Algorithm `Search_on_`$C(n,\mathbf{a},\mathbf{b},\rho^U)$
BEGIN
1. Set $p_i = 2, i = 1, 2, .., n$.
2. Set $C_i = a_i + b_i, i = 1, 2, .., n$.
3. Set $C = \max_i C_i$.
4. Set $\rho = \sum_{i=1}^{n} \frac{1}{p_i}$.
5. While $\rho > \rho^U$
   BEGIN
   6. Set $p_i = \left\lfloor \frac{C - a_i}{b_i} \right\rfloor + 1, i = 1, 2, .., n$.
   7. Set $C'_i = a_i + p_i b_i, i = 1, 2, .., n$.
   8. Set $C' = \min_i C'_i$.
   9. Update $\rho = \sum_{i=1}^{n} \frac{1}{p_i}$.
   10. If $\rho > \rho^U$, then update $C = C'$.
   END.
END.

Fig. 2. Pseudo-code for Algorithm `Search_on_`$C(n,\mathbf{a},\mathbf{b},\rho^U)$

This is a direct extension of the conjecture on the schedulability of pinwheel instances. Therefore, its proof does not exist in the literature and is out of the scope of this chapter.

At this point, we focus on obtaining a valid upper bound on C^*, based on a special type of periodic scheduling problem in the context of just-in-time (JIT) manufacturing. An ultimate goal of the JIT philosophy is to manufacture products at the exact time of demand, and, thus, minimize the costs associated with carrying inventories as well as backlogging or losing orders. Since the exact time of demand cannot be known in advance, demand is assumed uniformly distributed over the planning horizon. Accordingly, an ideal manufacturing schedule would produce each product in the exact rate of its demand. For example, if demand is expected to be 10 units for a given product in a 30-day horizon, then we should produce one unit every three days. For more on the JIT scheduling problem, we refer the reader to [6, 7, 10]

Steiner and Yeomans [8] address the JIT scheduling problem and prove that there always exists a sequence in which the ideal and actual cumulative production quantities of a product differ by at most one. Here, each product has a demand d_i in the planning horizon. The total demand $D = \sum_i d_i$ defines the length of the sequence, and, hence, the length of the planning horizon. Ideal cumulative production quantity up to stage k is defined by kd_i/D. Actual cumulative production quantity is the number of units of a product sequenced in the first k stages.

The JIT scheduling problem is similar to the sensor and pinwheel scheduling problems in structure, that is, the goal of evenly spacing products/sites/tasks over the sequence is common to all. Building on this point, we define a period $p_i = D/d_i$ for the production of i. Steiner and Yeomans's result [8] shows that there always exists a sequence that produces exactly one unit of product i in stages $(r-1)p_i + 1, .., rp_i$, for all $r = 1, 2, .., d_i$. Revisiting the above example, this result means that exactly one unit is produced in stages 1-3, one in 4-6, and

so forth. Here, note that sequencing the product in stages $1-6-7-12-\ldots$ is possible; we relate this result to the sensor scheduling problem as follows.

Lemma 1. *For an instance of the sensor scheduling problem, if* $\mathbf{p}$ *is the vector of periods obtained using* C^L*; then a sequence always exists such that two consecutive visits to site* i *are at most* $2p_i - 1$ *time steps apart.*

Proof. Let LCM be the least common multiplier of $p_1, p_2, .., p_n$. We can create an instance of the JIT scheduling problem by creating n products each with a demand of $d_i = LCM/p_i$. The summation of the demands may be less than LCM, in which case the gap should be filled by creating dummy products with a demand of 1 so that the dummies do not have an effect on the sequence. Through JIT scheduling, one can obtain a sequence where product i is produced exactly once in stages $(r-1)p_i + 1, .., rp_i$ for all $r = 1, 2, .., d_i$ and $i = 1, 2, .., n$. Also note that both d_i and p_i are integers. The largest possible distance between the positions r and $(r+1)$th copies of product i is $(r+1)p_i - ((r-1)p_i + 1) = 2p_i - 1$, for $r = 1, 2, .., d_i - 1$. □

Theorem 4. $C^U = 2C^L - \min_i a_i$ *is a valid upper bound on* C^*.

Proof. Given a C^L, we obtain the periods for each site with $p_i = \left\lfloor \frac{C^L - a_i}{b_i} \right\rfloor + 1$. From Lemma 1, we know that we can always find a sequence in which two consecutive visits to site i are at most $2\left\lfloor \frac{C^L - a_i}{b_i} \right\rfloor + 1$ time steps apart. Therefore the cost of information loss for site i is $C_i = a_i + (2\left\lfloor \frac{C^L - a_i}{b_i} \right\rfloor) b_i \leq a_i + 2\left\lfloor C^L - a_i \right\rfloor = 2C^L - a_i$. □

5 Heuristic Solution Approaches

The sensor scheduling problem is NP-Hard as shown earlier in this chapter. Furthermore, infinitely long sequences are sought as complete solutions to the problem. These two facts render exact solution methods impractical. Therefore, developing time-efficient constructive heuristic procedures is beneficial.

A constructive heuristic starts with a null solution, which is an empty sequence in our case. Recalling decision variable $y_{i,t}$ of our optimization model, we assume $y_{i,0} = 0$ for all $i = 1, 2, .., n$. In other words, it is assumed that at the beginning, we have perfect information about all sites. In each stage, exactly one site is visited, and, hence, there is exactly one i satisfying $y_{i,t} = t$ $(t = 1, 2, \ldots)$. For the $n-1$ sites not visited in stage t, we have $y_{i,t} = y_{i,t-1}$. Now we define a time since the last visit to site i by stage t: $z_i(t) = t - y_{i,t}$. Note that in each stage exactly one $z_i(t) = 0$ and the remaining $n-1$ are positive integers. Moreover, $z_i(t)$ increases in t until site i is visited.

As discussed earlier in the chapter, for a given C, we can derive visit periods p_i for each site and reduce the problem to an instance of the pinwheel scheduling problem. If this instance is schedulable, then $z_i(t) \in \{0, 1, .., p_i\}$, for all $i = 1, 2, .., n$ and $t = 1, 2, \ldots$ Therefore, the number of different values the vector

$\mathbf{z}(t)$ can take is finite. This result implies that after a finite number of steps, the $\mathbf{z}(t)$ vector will repeat itself. That is, if we can identify such a stage, we can build a cyclic sequence by repeating the stages between consecutive occurrences of the same $\mathbf{z}(t)$. This result constitutes a main principle used in both our heuristic procedures. Another common principle is based upon Theorem 2, prohibiting the sensor from staying focused on the same site in two consecutive stages. In other words, our constructive heuristics evaluate all sites but the one that has been just visited for the next move, in all stages.

Our first constructive heuristic is a greedy procedure. It starts with the initial visit history as described above ($\mathbf{z}(0) = \mathbf{0}$). It evaluates all n sites that can be visited in the first stage. The selection of the site to visit is made to minimize the penalty of information loss, i.e., penalty of not visiting a site. After the selection the visit history is updated. Note that in the later stages the method evaluates $n-1$ sites for its next visit. Repeating this simple selection and update operations until a repetition in the visit history is observed constitutes the framework of our greedy heuristic. We improve its performance by adding a look-ahead feature. The heuristic still evaluates $n-1$ possible sites to visit in each stage, but makes the decision based on the cost observed in the next ℓ stages. Larger ℓ values are expected to yield better (lower cost) solutions on the average, however it is not guaranteed. On the other hand, the number of operations to perform increases with ℓ, rendering small ℓ values more computationally efficient. We call our first heuristic *greedy with look-ahead* (GLA).

Our second heuristic is an alternative greedy approach that dynamically sets a deadline to visit each site and then selects the site with the earliest deadline for the sensor's next move. More specifically, it starts with a small C and calculates periods p_i for each site to achieve that C value. For each $i = 1, 2, .., n$ and $t = 1, 2, ..$, the deadline for the next visit to site i is set to $y_{i,t-1} + p_i$. Then, the site with the earliest deadline is selected for the next visit (ties can be broken arbitrarily). However, if the C value at hand is too small, then in some stage the method will unavoidably have more than one site that must be visited in that stage. Since this is infeasible, the method increases C until at most one site must be visited in that stage. The termination again is based on observing a repetition in the visit history. We call this heuristic *dynamic deadlines* (DD).

6 Computational Study

We consider four different numbers of sites: $n \in \{4, 6, 8, 10\}$. The number of sites can also be considered the problem size. For each problem size, we pseudo-randomly create 100 test instances with $b_i \in \{1, .., 10\}$ and $a_i \in \{11, .., 100\}$. Therefore, we have a total of 400 test instances.

In this study, for each instance, we first obtain C^L, C^U and C^C. Then, we run the two heuristics. From our preliminary experiments we have observed that $\ell = n$ works best. Therefore, we run the GLA method with $\ell = n$ only.

We know that C^U is always greater than C^L and less than $2C^L$. However, we cannot make such clear inferences about C^C. Therefore, we are interested in

the relative position of C^C to C^L and C^U. We calculate the relative position with $(C^C - C^L)/(C^U - C^L)$. Similarly, we calculate the relative positions of the solutions obtained by the GLA and DD heuristics, as well. For example, if the relative position of the GLA method's solution is calculated as 0.25, we understand it is located at 25% of the distance from the lower bound to the upper bound. In other words, smaller values represent better solutions. C^L, C^U and C^C are computed almost instantly, thus we are not concerned about their time consumption. The heuristic methods, on the other hand, can take a significant amount of computation time depending on the problem size. The results are summarized in Figures 3 and 4.

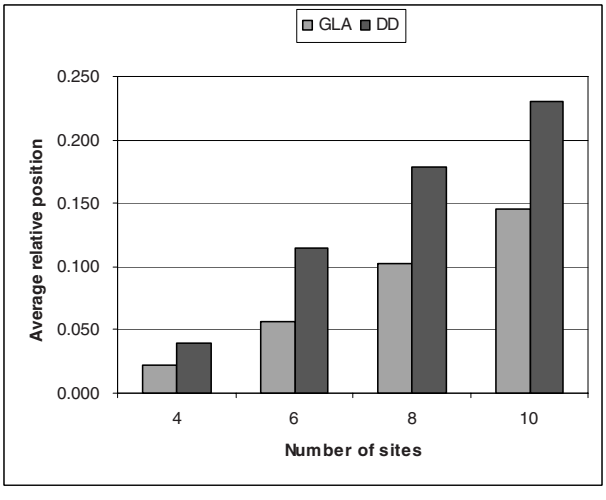

Fig. 3. Relative positions

Our two heuristics perform well in general as their relative position is always closer to the lower bound than the upper bound. Hence, we state that our methods are effective in solving the sensor scheduling problem. When the two heuristics are compared, we see that GLA outperforms DD on all problem sizes. Computation time of both methods increases significantly with problem size, DD taking longer than GLA. Thus, we state that GLA heuristic is superior to DD. Even so, the results show that both methods are computationally efficient in that they solve the problem in seconds.

The conjectured upper bound is found to be tighter than the valid upper bound developed in this chapter. Furthermore, the conjectured upper bound seems to work better than the heuristic methods on larger problem sizes. However, a sequencing procedure is not known in the existing literature to support the conjecture. Therefore, with their negligible computational burden and high performance, the heuristics proposed in this chapter can be used to solve the sensor scheduling problem in practice.

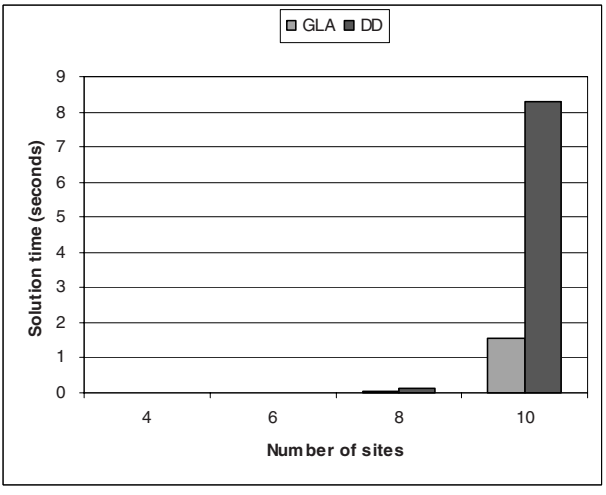

Fig. 4. Computation times

7 Conclusions

This chapter addresses the scheduling problem of a sensor that constantly collects information from multiple sites. In the earlier work, the problem is solved by probabilistic approaches, potentially generating schedules in which a site is not visited for a long time. To overcome this deficiency, this chapter presents a deterministic approach formulated as an integer linear program. Upon showing that the problem is NP-Hard, the chapter develops valid lower and upper bounds and proposes two constructive heuristic methods. Tested via an extensive computational study, the heuristic methods are efficient and effective in solving the problem.

The results also pinpoint the need to prove the widely believed "5/6" conjecture of the pinwheel scheduling literature and to develop efficient algorithms to solve the pinwheel scheduling problem. In the existing literature, algorithms developed for the pinwheel problem either require a small number (2-3) of distinct periods or have low density guarantees. A comprehensive scheduling method for the pinwheel is therefore critical for the solution of the sensor scheduling problem in the general case.

The problem studied in this chapter belongs to a rich and relatively unexplored area. Promising future research directions in the area include multiple sensors in a cooperative framework and non-unit switch-over/observation times between the sites, with a combination of the two being the ultimate goal. Also, investigation of the problem under time-variant site dynamics, and comparison of the deterministic heuristic procedures with probabilistic approaches are possible future research directions.

Bibliography

[1] A. Bar-Noy, R. Bhatia, J.S. Naor, and B. Schieber. Minimizing service and operations costs of periodic scheduling. *Mathematics of Operations Research*, 27:518–544, 2002.

[2] D. Chen and A. Mok. *The pinwheel: A real-time scheduling problem*, chapter 27. Chapman & Hall/CRC, 2004.

[3] E.A. Feinberg and M.T. Curry. Generalized pinwheel problem. *Mathematical Methods of Operations Research*, 62:99–122, 2005.

[4] J. Filar. Player aggregation in the traveling inspector model. *IEEE Transactions on Automatic Control*, 30:723–729, 1985.

[5] J.A. Filar and T.A. Schultz. The traveling inspector model. *OR Spectrum*, 8:33–36, 1986.

[6] J. Miltenburg. Level schedules for mixed-model assembly lines in just-in-time production systems. *Management Science*, 35(2):192–207, Feb. 1989.

[7] Y. Monden. *Toyota Production System: An Integrated Approach to Just-In-Time*. Engineering & Management Press, third edition, 1998.

[8] G. Steiner and S. Yeomans. Level schedules for mixed-model, just-in-time processes. *Management Science*, 39(6):728–735, June 1993.

[9] A. Tiwari, M. Jun, D. Jeffcoat, and R. Murray. The dynamic sensor coverage problem. In *Proceedings of the 16th International Federation of Automatic Control (IFAC) World Congress*, Prague, Czech Republic, July 2005.

[10] M. Yavuz and E. Akcali. Production smoothing in just-in-time manufacturing systems: A review of the models and solution approaches. *International Journal of Production Research*, 2007. Forthcoming.

[11] N. Yerrick, A. Tiwari, and D. Jeffcoat. An investigation of a dynamic sensor motion strategy. In *Proceedings of the 6th Cooperative Control and Optimization Conference*, Gainesville, FL, February 2006. World Scientific.

[12] N. Yerrick, M. Yavuz, and D. Jeffcoat. Two sensor motion models for the dynamic sensor coverage problem. *Military Operations Research*, 2007. Forthcoming.