Radar Pulse Interleaving for Multi-target Tracking

⋆ Moustafa Elshafei

⋆⋆ Hanif D. Sherali

⋆⋆⋆ J. Cole Smith

⋆ Department of Systems Engineering
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia

⋆⋆ Grado Department of Industrial and Systems Engineering (0118)
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

⋆⋆⋆ Department of Systems and Industrial Engineering
University of Arizona
Tucson, AZ 85721

Acknowledgment: Dr. M. Elshafei acknowledges the King Fahd University of Petroleum and Minerals for its support in conducting this research. Dr. H.D. Sherali also acknowledges the support of the *National Science Foundation* under Grant Nos. DMI-9812047 and DMI-0085640.

Abstract

In a multifunction radar, the maximum number of targets which can be managed or tracked is an important performance measure. Interleaving algorithms developed to operate radars exploit the dead-times between the transmitted and the received pulses to allocate new tracking tasks that might involve transmitting or receiving pulses, thus increasing the capacity of the system. The problem of interleaving N targets involves a search among $N!$ possibilities, and sub-optimal solutions are usually employed to satisfy the real-time constraints of the radar system. In this paper, we present new tight 0-1 integer programming models for the radar pulse interleaving problem, and develop effective solution methods based on Lagrangian relaxation techniques.

Key Words: Radar tracking, pulse interleaving, 0-1 integer programming, Lagrangian relaxation.

1 Introduction

A multifunction radar, which is based on an electronically phased array, can steer the antenna beam toward any direction almost instantaneously [13]. Because of this feature, a multifunction radar can simultaneously perform two kinds of tasks: surveillance and tracking. Air defense radars may also be required to perform additional tasks, including missile midcourse guidance and target-via-missile (TVM) operations. Such tasks are used for example, in the Patriot air defense system [2, 8].

The last decade has witnessed a significant enhancement in the technology of ballistic missiles (BM) and atomic warheads, which pose a serious challenge for defense systems. Further technologies such as the Multiple Independently Targeted Re-entry Vehicle (MIRV) that can dispatch several nuclear warheads from a single in-flight missile, and the Russian-built fractional orbit bombardment system (FOBS) that permits missiles or warheads to remain in earth orbit before beginning their descent, also represent a serious threat to defense systems. Such an FOBS technology provides the ability to launch a massive attack from any direction rather than just depending on a ballistic pathway arching over the North Pole. The task of tracking these multi-targets can be even harder in the presence of debris, chaff, decoys, and Electronic Counter Measures (ECM). Due to the enormous speed of the BM at re-entry (6-10 km/s), even with the current satellite early warning systems, the radar defense system could be left with only 10-20 seconds from spotting the re-entering objects to identifying the true targets, tracking these targets, and engaging counter-defense actions.

In such scenarios, the enormous speed of the approaching targets necessitates a high updating rate of the target tracking information, including the estimation of target position and velocity, in order to facilitate an effective operation of the anti-missile defense system. It is then vital to employ efficient operational methods that would minimize the cycle time required to update the information for all possible targets.

On the other hand, in conventional air defense systems and in air traffic control, the critical performance measure is the ability of the radar system to track a large number of targets or aircraft. Once the surveillance tasks have been defined, the objective is to maximize the number of targets that are tracked during an available time horizon (total time minus

the surveillance time). In all these applications, since we have an estimate of the range (distance) of the targets, we know the dead-time between the transmitted pulse and the received (reflected) pulse within a specific range of uncertainty. Therefore, we can interleave other pulses in these intervals in order to enhance the effectiveness and the timeliness of the tracking operations.

The computational burden of scheduling the transmitting of radar pulses for N targets increases with the number of targets, being governed by the search among $N!$ possibilities, and usually, sub-optimal solutions are employed [3, 5]. These solutions could be obtained by sorting the targets based on some criterion as in the Farina and Neri algorithm [3], which orders the targets according to decreasing values of their distances from the radar, and then allocates each pulse to a time slot in a feasible manner using this ordered scheme. In another approach [5], the authors propose the use of a Hopfield Neural Network for a scenario that assumes no uncertainties in the received pulses.

In this paper, we investigate efficient solution methods for the radar pulse interleaving problem based on Lagrangian relaxation approaches to new 0-1 integer programming models of the problem that possess tight continuous relaxations [9]. Nemhauser and Wolsey [7] provide a general discussion on Lagrangian relaxation, and references [11, 6] discuss models related to similar task scheduling problems. We formulate the radar pulse interleaving problem using two possible objective functions as follows.

1. **Problem 1.** Given a fixed time interval T, and given N targets, each with an associated risk (cost) of remaining unscanned, how can we scan the targets to minimize the overall risk of the unscanned targets?

2. **Problem 2.** Given N targets, what is the minimum time required to scan all of them?

In Section 2, we describe these two problems in more detail and discuss their modeling assumptions. We mathematically formulate these problems in Section 3 and develop several specialized techniques for their solution in Sections 4 and 5, respectively. In Section 6, we provide some results and compare the effectiveness of the proposed methods versus the heuristic described in [3]. Finally, we conclude this paper in Section 7, providing a summary of our results and possible extensions for future research.

2 Problem Description

Let us define the following (see Figure 1).

T: number of time slots, each of duration T_s seconds.

τ_i: transmit pulse duration (integral number of time slots) for the ith target.

s_i: the starting time slot of the transmit pulse to track the ith target.

r_i: the time slot at the center of the duration over which the radar receiver should be on to completely receive the echo pulse of the ith target.

Δ_i: uncertainty measure for the received pulse, whereby $2\Delta_i + 1$ represents the duration (integral number of time slots) over which the radar receiver should be on to completely receive the echo pulse of the ith target, (i.e., Δ_i on either side of slot r_i). The received echo pulse is assumed to be totally contained in this period, $2\Delta_i + 1$, as depicted in Figure 1.

d_i: the estimated gap (integral number of time slots) between s_i and r_i (governed by the distance to the target, and the receiving pulse period) as ascertained from the previous scans. (Note that $d_i \geq \tau_i + \Delta_i$.)

The receiving pulse uncertainty, Δ_i, depends on τ_i, the target speed, the radar cell size, the target tracking cycle length, the range of the target, and the number of targets to be tracked. For example, using simple mathematics, if the target travels at a radial speed of 600 m/s and the tracking cycle is 1.0 second, the target will be displaced by 600 meters from the previous scan cycle, and the new echo pulse gap could be different from d_i by 4.0 μ-seconds. Clearly, Δ_i depends on the allowable uncertainty in the target position, which is expected to be small for nearby targets, but much more tolerable for far away targets.

The transmit pulse duration τ_i is different from the true transmit pulse width, where the latter can be between 0.1 μ-seconds for high resolution radars to more than 100 μ-seconds for conventional radars. The choice of this pulse duration depends on the target range, the required resolution, and the dwell time. Alternatively, τ_i could represent a duration during which multiple radar pulses are transmitted. In many cases, it is desirable to illuminate the target using a series of consequent pulses to mitigate the effect of clutter or slowly moving objects. Multiple pulses with carrier frequency shifts are also employed for Electronic

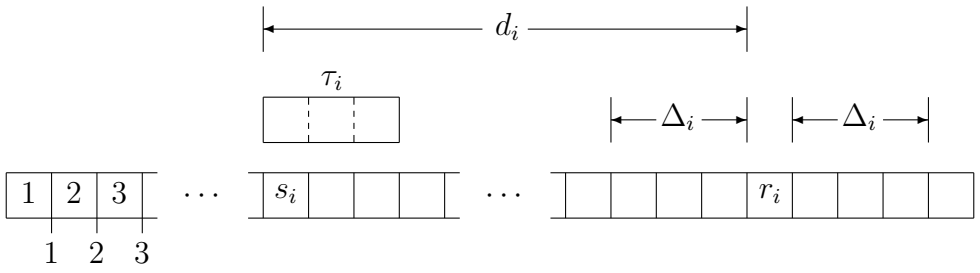

Figure 1: Illustration of the transmitted pulse, the received pulse, and the uncertainty for the ith target.

Counter-Counter Measures (ECCM). In addition, τ_i depends also on the transmitter maximum duty cycle, which dictates a certain waiting time before the next radar transmitter pulse is dispatched. In high performance solid state radars, the duty cycle is between 0.02 to 0.1 μ-seconds, while in conventional radars, it varies between 0.2-0.5 μ-seconds.

In this paper, we are concerned with the basic task of scheduling a transmit pulse of width τ_i for the ith target, and we assume that τ_i is a given entity based on the radar duty cycle and the type of radar pulse that has been designated for this target. It is required to determine the schedule for all the transmitted and received pulses, ensuring that each time slot is occupied by at most one task.

Example 1. Suppose that we are given $T = 12$ and five targets having d_1=2, d_2=3, d_3=4, d_4=5, and d_5=6. For simplicity, if we assume Δ_i=0 and τ_i=1 for all i, then one possible arrangement is shown in Table 1. However, it is not clear if this solution is optimal or not. Note that there are two unused time slots (slots 7 and 11), yielding a time utilization efficiency of (10/12), or about 83%.

As a quick estimate, it is readily seen that the minimum number of time slots, M, to interleave radar pulses for N targets satisfies

$$\sum_{i=1}^{N} \tau_i + N + 2\sum_{i=1}^{N} \Delta_i \leq M \leq N + \sum_{i=1}^{N} (d_i + \Delta_i). \tag{1}$$

1	2	3	4	5	6	7	8	9	10	11	12
									s_1		r_1
		s_2			r_2						
s_3				r_3							
			s_4					r_4			
	s_5						r_5				

Table 1: Feasible solution for Example 1.

The left-hand side of (1) is based on assuming a 100% utilization of the total number of time slots, and the right-hand side is based on performing the tasks one after the other in any order. In this case, the transmitted pulse for each task is sent and the radar waits until it receives its echo pulse before transmitting the next pulse.

3 Model Formulations

Let us define the set $\boldsymbol{T} = \{1, ..., T\}$, and the binary variables x_{ij} as

$$\begin{aligned} x_{ij} &= 1 \qquad \text{if the transmit pulse of target } i \text{ begins at slot } j \in \boldsymbol{T} \\ x_{ij} &= 0 \qquad \text{otherwise.} \end{aligned} \tag{2}$$

Note that if we examine any slot k, then this slot is occupied by the task of either transmitting or receiving pulses with respect to target i, if either $x_{ij} = 1$ for $j \in \{k - \tau_i + 1,...,k\} \cap \boldsymbol{T}$ or if $x_{ij} = 1$ for $j \in \{k - d_i - \Delta_i,...,k - d_i + \Delta_i\} \cap \boldsymbol{T}$. (The intersection with $\boldsymbol{T}$ ensures that only defined values of j are considered.) Accordingly, let us define

$$J_k = \{(i, j) : \text{ if } x_{ij} = 1, \text{ then slot } k \text{ would be occupied}\}, \text{ for all slots } k = 1, ..., T.$$

From the previous statement, we have that

$$\begin{aligned} J_k = \{(i, j) : \ & \text{target } i \in \{1, ..., N\}, \text{ with either } j \in \{k - \tau_i + 1, ..., k\} \cap \boldsymbol{T} \text{ or} \\ & j \in \{k - d_i - \Delta_i, ..., k - d_i + \Delta_i\} \cap \boldsymbol{T}\} \quad \text{for } k = 1, ..., T, \end{aligned} \tag{3}$$

where undefined slots j in (3) are omitted from consideration. Furthermore, let C_i denote the cost or risk of missing (not pulsing) target i. Hence, our objective is to minimize the total cost given by (noting that $\sum_j x_{ij} \leq 1 \; \forall \; i$)

$$J = \sum_{i=1}^{N} C_i \left[1 - \sum_{j=1}^{t_i} x_{ij} \right] = \sum_{i=1}^{N} C_i - \sum_{i=1}^{N} \sum_{j=1}^{t_i} C_i x_{ij}, \tag{4}$$

where $t_i \equiv T - d_i - \Delta_i$, $\forall \; i = 1, ..., N$ are introduced in order to ensure that if the target i is pulsed by t_i ($x_{it_i} = 1$), then in the worst case, the echo pulse will have been received by the end of slot T. Consequently, based on (4), we can equivalently state the problem of minimizing the total risk subject to the pulse or task constraints as follows.

$$\textbf{P1: Maximize} \sum_{i=1}^{N} \sum_{j=1}^{t_i} C_i x_{ij} \tag{5a}$$

$$\text{subject to} \sum_{j=1}^{t_i} x_{ij} \leq 1, \quad \forall \; i = 1, ..., N \tag{5b}$$

$$\sum_{(i,j) \in J_k} x_{ij} \leq 1, \quad \forall \; k = 1, ..., T \tag{5c}$$

$$x_{ij} \in \{0, 1\} \quad \forall \; i = 1, ..., N, \; j = 1, ..., t_i. \tag{5d}$$

The objective function (5a) is equivalent to minimizing (4), constraints (5b) require each target to be pulsed at most once, constraints (5c) enforce that no tasks overlap during any of the time slots, and constraints (5d) are logical restrictions on the defined decision variables.

For modeling Problem 2, we can first apply a simple heuristic such as that described in [3] to find a feasible solution of total duration T slots, say, and then define the problem of minimizing the makespan on the time slots spanning this duration. Noting that the completion time (when the final pulse is received) for target i can be written as

$$\sum_{j=1}^{t_i} (j + d_i + \Delta_i) x_{ij} \quad \forall \; i = 1, ..., N, \tag{6}$$

we can formulate P2 as follows, where the objective function variable z represents the makespan.

$$\textbf{P2:}\ \text{Minimize } z \tag{7a}$$

$$\text{subject to } z \geq \sum_{j=1}^{t_i}(j + d_i + \Delta_i)x_{ij} \quad \forall\ i = 1, ..., N \tag{7b}$$

$$\sum_{j=1}^{t_i} x_{ij} = 1, \quad \forall\ i = 1, ..., N \tag{7c}$$

$$\sum_{(i,j)\in J_k} x_{ij} \leq 1, \quad \forall\ k = 1, ..., T \tag{7d}$$

$$x_{ij} \in \{0, 1\} \quad \forall\ i = 1, ..., N,\ j = 1, ..., t_i. \tag{7e}$$

Note that in contrast with (5b), since each target is now designated to be pulsed, constraints (7c) are written as equality restrictions.

Remark 1. P1 is a set packing problem and is likely to have a tight linear programming relaxation. This could be further enhanced using the Reformulation-Linearization Technique (RLT) as in Sherali, Adams and Driscoll [9] based on pairwise constraint products. In contrast, even though the partitioning-packing structure (7c)-(7e) might have a tight linear programming relaxation, this is not likely the case in the added dimension of z, along with the constraints (7b). In other words, the problem of minimizing a piecewise linear convex function over the region defined by (7c)-(7e) is likely to yield a weak lower bound when the integrality conditions (7e) are relaxed. Hence, we expect P2 to be a significantly more difficult problem to solve to (near) optimality. In light of this, as an alternative to using P2, we reflect the philosophy of completing all the tasks as early as possible by penalizing late completions at an increasing (squared) rate. For example, we could formulate the following problem, where T is determined as for P2.

$$\textbf{P2}'\textbf{:}\ \text{Minimize } \sum_{i=1}^{N}\sum_{j=1}^{t_i}(j + d_i + \Delta_i)^2 x_{ij} \tag{8a}$$

$$\text{subject to (7c)} - \text{(7e)}. \tag{8b}$$

In our computations, we study the effect of solving this problem in lieu of P2 on determining the minimal makespan.

Another alternative in this same vein is to solve the linear programming relaxation of P2, and then compose an objective function for use in P2′ by surrogating (7b) using the optimal dual variable values obtained from this linear programming solution. Let $\overline{\lambda}_i \; \forall \; i \in N$ denote the optimal values of the dual variables associated with (7b), and define

$$C_{ij} = \overline{\lambda}_i(j + d_i + \Delta_i) \qquad \forall \; i = 1, ..., N, \; \forall \; j = 1, ..., t_i. \tag{9}$$

We may then solve the following problem.

$$\textbf{P2}'': \text{ Minimize } \sum_{i=1}^{N} \sum_{j=1}^{t_i} [C_{ij} + (j + d_i + \Delta_i)^2] x_{ij} \tag{10a}$$

$$\text{subject to (7c)} - \text{(7e)}. \tag{10b}$$

We also investigate this variant for designing a heuristic method for Problem P2. A theoretical study that explores the connections between P2 and its alternatives P2′ and P2″ is an interesting topic of research that we propose for future study.

4 Lagrangian Relaxation Approach for Problem 1

In this section, we develop a Lagrangian dual technique for providing tight upper and lower bounds within a branch-and-bound framework for solving Problem P1. Let u_k, $k = 1, ..., T$ be Lagrangian multipliers associated with (5c). We can then construct a Lagrangian dual formulation for Problem P1 as follows.

$$\textbf{LDP1:} \text{ Minimize } \{v(u) : u \geq 0\}, \tag{11}$$

where $v(u)$ is evaluated via the following Lagrangian subproblem, given multipliers $u \equiv (u_k, \forall\ k \in T)$.

$$v(u): \text{ Maximize } \sum_{i=1}^{N}\sum_{j=1}^{t_i} C_i x_{ij} + \sum_{k=1}^{T} u_k \left(1 - \sum_{(i,j)\in J_k} x_{ij}\right) \tag{12a}$$

$$\text{subject to } \sum_{j=1}^{t_i} x_{ij} \leq 1, \quad \forall\ i = 1, ..., N \tag{12b}$$

$$x \geq 0. \tag{12c}$$

Note that (12) is trivially solvable, yielding a binary extreme point optimum. This property also entails that the optimal value of LDP1 equals that of the linear programming relaxation to Problem P1. Consider the following heuristic approach for solving this Lagrangian dual problem, using the primal solution recovery procedure of Sherali and Choi [10].

Heuristic HLDP1 for Solving LDP1

Initialization: Let N_1 be a constant representing the maximum number of iterations permitted, let ϵ_1 and ϵ_2 be suitable termination tolerances, let N_2 be the number of iterations performed before triggering the primal solution recovery scheme, and let N_3 be the minimum number of iterations between successive primal heuristic applications. Initialize the iteration (or step) counter $s = 1$. Obtain a lower bound $\overline{z}$ (> 0) for Problem LDP1 by executing the bounding heuristic described in Section 4.4.1 below, and let the initial Lagrangian multipliers be $u_k^s = 0$ for $k = 1, ..., T$. Initialize the incumbent dual upper bound value to $\hat{z} = \infty$, and the aggregate primal recovery solution $\tilde{x} \equiv 0$.

Step 1: Solve the subproblem (12) to evaluate $v(u^s)$, and obtain the corresponding solution $\overline{x}^s$. Let $z^s \equiv v(u^s)$, and if $s > N_2$, set $\tilde{x} \leftarrow \tilde{x} + \overline{x}^s$. Compute the subgradient g^s of $v(\cdot)$ at $u = u^s$ having components $g_k^s = \left(1 - \sum_{(i,j)\in J_k} \overline{x}_{ij}^s\right)$, $k = 1, ..., T$. If $z^s < \hat{z}$, then set $\hat{z} = z^s$ and store u^s as the incumbent dual solution. If $s > N_2$ in addition to the previous condition $z^s < \hat{z}$, determine the current primal estimate $\hat{x} = \tilde{x}/(s - N_2)$, and compute an updated lower bound $\overline{z}$ based on the heuristic described in Section 4.4.4 below, provided at least N_3 iterations have been executed since last invoking this heuristic. (This latter check tends to ensure that $\hat{x}$ has changed sufficiently enough to make a difference within the heuristic scheme.)

Step 2: Compute a direction d^s and a step size λ^s for updating the Lagrangian multipliers u. (Appropriate choices for d^s and λ^s are given in Sections 4.4.2 and 4.4.3, respectively.) Set $u^{s+1} = P_{u\geq 0}[u^s + \lambda^s d^s]$, where $P_{u\geq 0}[\cdot]$ projects $[\cdot]$ onto $\{u\colon\ u \geq 0\}$. Increment s by 1. If $s > N_1$, $(\hat{z} - \overline{z})/\overline{z} \leq \epsilon_1$, or $||g^s|| \leq \epsilon_2$, then terminate. Otherwise, return to Step 1.

4.4.1 Initial Lower Bounding Heuristic

To obtain a lower bound $\overline{z}$ for LDP1 via an initial feasible solution for P1, we use the following greedy algorithm, which attempts to schedule pulsing for the most critical (highest C_i valued) jobs first before proceeding to less critical jobs. We denote this heuristic **HP1**.
Initialization: Assume that all jobs $i = 1, ..., N$ have been sorted in nonincreasing order of their C_i-values. Let $F = 1$ denote the first available slot in the horizon. Initialize the iteration counter $i = 1$.
Step 1: If any slot $F, ..., F + \tau_i - 1$ or $F + d_i - \Delta_i, ..., F + d_i + \Delta_i$ is unavailable, then proceed to Step 2. Otherwise, begin the pulsing of target i at F. Update $F \leftarrow F + \tau_i$ and increment i by one. If $i > N$, stop. Otherwise, repeat Step 1.
Step 2: Find the smallest $r \geq 1$, if it exists, such that $F + r, ..., F + r + \tau_i - 1$ and $F + r + d_i - \Delta_i, ..., F + r + d_i + \Delta_i$ are unoccupied, where $F + r + d_i + \Delta_i \leq T$. If r exists, schedule the pulsing of target i at $F + r$. In any case, increment i by one. If $i > N$, stop. Otherwise, return to Step 1. (Note that F is *not* updated at this step, thereby permitting other subsequent targets to be possibly pulsed prior to the target(s) considered thus far.)

4.4.2 Method for Selecting the Direction d^s

We employ Sherali and Ulular's [12] average direction strategy (ADS), in which the new direction for each iteration bisects the angle between the previous direction and the subgradient direction. At $s = 1$, $d^s = -g^s$. For $s \geq 2$, we take

$$d^s = -g^s + \frac{||g^s||}{||d^{s-1}||} d^{s-1}. \tag{13}$$

4.4.3 Method for Selecting the Step Size λ^s

In selecting our step size λ^s for each iteration, we combine ideas from Held et al. [4] and Sherali and Ulular [12]. Held et al. [4] prescribe a step length given by

$$\lambda^s = \beta^s \frac{(z^s - \overline{z})}{||d^s||^2}, \tag{14}$$

where β^s is a suitable parameter, and $\overline{z}$, z^s, and d^s are defined above. To accelerate the procedure, we used the block-halving scheme of Sherali and Ulular [12]. In this strategy, we divide the N_1 maximum allowable iterations into several blocks of N_{1b} iterations each. Within each of these blocks, we initialize the step length using (14), with $\beta^s \equiv \overline{\beta}/\lceil s/N_{1b} \rceil$ for some specified $\overline{\beta}$, and then we retain this initial step length throughout the corresponding block.

4.4.4 Lower Bounding Heuristic at Step 1 of HLDP1

From the Lagrangian procedure, we have a current primal solution estimate $\hat{x}$. For $i = 1, ..., N$, let $y_i \in \arg\max_{j=1,...,t_i}\{\hat{x}_{ij}\}$. Consider the solution given by pulsing each target i at time slot y_i, whenever $\hat{x}_{iy_i} > 0$. This solution is likely to be infeasible. Define f_k, $k = 1, ..., T$, to be the degree of infeasibility for each time slot k, that is, the number of targets occupying slot k during their pulsing or receiving phases in *excess* of one. Let $F = \sum_{k=1}^{T} f_k$ be the total amount of infeasibility in the current solution. If no infeasibilities occur, we proceed directly to Step 8.

Step 1: For each target $i = 1, ..., N$ that is being pulsed in the current solution, examine the possible beginning pulsing times for i (from 1 to t_i excluding y_i) that would yield a decrease in F. Let M be the set of all such possible moves (i, y_i, y_i^l), where y_i^l would be the new value of y_i under move l. If $M = \emptyset$, go to Step 2. Otherwise, go to Step 3.

Step 2: Identify the scheduled pulse which is currently being pulsed or received during the most number of infeasible time slots (time slots occupied by two or more targets). Cancel the pulsing of this target, and go to Step 4.

Step 3: Find the move l^* in M which maximizes the decrease in F. Set $y_i = y_i^{l^*}$, and go to Step 4.

Step 4: Revise f and F based on the modified current solution. If $F = 0$, proceed to Step 5. Otherwise, return to Step 1.

Step 5: For each unscheduled target, check (in nonincreasing order of their C values) to see if the target can be feasibly interleaved among the currently scheduled pulses. Perform any such possible insertions and terminate.

Remark 2. We may also approach the solution of Problem 1 by equivalently reformulating P1 in the following manner in order to reveal a particular network substructure. Define variables

$$y_{ij} = \begin{cases} 1 & \text{if the receiving pulse blocked-off duration begins at slot } j \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for } i = 1, ..., N,\ j = 1 + d_i - \Delta_i, ..., T - 2\Delta_i, \tag{15}$$

where undefined y-variables are assumed to be zero below. Furthermore, in lieu of J_k, define

$$J_k^x = \{(i,j) : i \in \{1, ..., N\}, j \in \{k - \tau_i + 1, ..., k\} \cap \boldsymbol{T}\} \tag{16a}$$

and

$$J_k^y = \{(i,j) : i \in \{1, ..., N\}, j \in \{k - 2\Delta_i, ..., k\} \cap \boldsymbol{T}\}. \tag{16b}$$

Then we may transform (5) equivalently to the following problem.

$$\textbf{P1}'\textbf{:}\ \text{Maximize} \sum_{i=1}^{N} \sum_{j=1}^{t_i} C_i x_{ij} \tag{17a}$$

$$\text{subject to} \sum_{j=1}^{t_i} x_{ij} \leq 1, \quad \forall\ i = 1, ..., N \tag{17b}$$

$$\sum_{(i,j) \in J_k^x} x_{ij} + \sum_{(i,j) \in J_k^y} y_{ij} \leq 1, \quad \forall\ k = 1, ..., T \tag{17c}$$

$$x_{ij} = y_{i,j+d_i-\Delta_i} \quad \forall\ i = 1, ..., N,\ j = 1, ..., t_i \tag{17d}$$

$$x_{ij} \in \{0, 1\} \quad \forall\ i = 1, ..., N,\ j = 1, ..., t_i, \text{ and}$$

$$y_{ij} \geq 0 \quad \forall\ i = 1, ..., N,\ j = 1 + d_i - \Delta_i, ..., T - 2\Delta_i. \tag{17e}$$

Note that constraints (17c) have the *interval* matrix property, in which the coefficients are all 0's and 1's, and in each column, the 1's appear consecutively [7]. Such structures are reducible to a network structure by subtracting the first row from the second row, the second row from the third row, and so on. We attempted to exploit this feature within a Lagrangian optimization algorithm by dualizing constraints (17b) and (17d), and constructing a Lagrangian dual subproblem consisting of the network constraints derived from (17c). However, our computational experience demonstrated that solving the Lagrangian dual problem based on P1 yields significantly tighter bounds at termination of the proposed (heuristic) scheme than those achieved by similarly solving the Lagrangian dual based on P1′. We also found that directly solving P1′ by integer programming is inefficient as compared with optimally solving P1. □

5 Minimizing the Total Completion Time: Problem 2

Recall that Problem 2 described in Section 1 is concerned with minimizing the makespan, i.e., minimizing the latest completion time of all the tasks. We now develop a Lagrangian dual heuristic for solving Problem 2 similar to the one developed for Problem 1.

Let $\gamma_i,\ i = 1, ..., N$ be Lagrangian multipliers associated with (7b), and let $u_k,\ k = 1, ..., T$ be Lagrangian multipliers associated with (7d). We construct the Lagrangian dual formulation for Problem P2 as follows.

$$\textbf{LDP2: Maximize } \{v(\gamma, u) : \gamma \geq 0, e\gamma = 1, u \geq 0\}, \tag{18}$$

where e is a vector of N ones, and $v(\gamma, u)$ is evaluated via the following Lagrangian subproblem, given multipliers γ and u.

$$v(\gamma, u): \text{ Minimize } \sum_{i=1}^{N} \gamma_i \left(\sum_{j=1}^{t_i} (j + d_i + \Delta_i) x_{ij} \right) + \sum_{k=1}^{T} u_k \left(\sum_{(i,j) \in J_k} x_{ij} - 1 \right) \tag{19a}$$

$$\text{subject to } \sum_{j=1}^{t_i} x_{ij} = 1, \quad \forall\ i = 1, ..., N \tag{19b}$$

$$0 \leq x_{ij} \leq 1 \quad \forall\ i = 1, ..., N,\ j = 1, ..., t_i. \tag{19c}$$

The problem (19) is trivially solvable, and because of its integrality property, the optimal value of LDP2 is equal to that of the linear programming relaxation to P2.

Heuristic HLDP2 for Solving LDP2

Initialization: Let N_1, N_2, N_3, ϵ_1, and ϵ_2, be algorithmic parameters as defined in HLDP1. Initialize the iteration counter $s = 1$. Obtain an upper bound $\overline{z}$ for Problem LDP2 by executing the heuristic of Farina and Neri [3] described in Section 5.5.1, and let the initial Lagrangian multipliers $\gamma_i^s = 1/N$ for $i = 1, ..., N$, and $u_k^s = 0$ for $k = 1, ..., T$. Initialize the incumbent dual lower bound value to $\hat{z} = -\infty$, and the aggregate primal recovery solution to $\tilde{x} \equiv 0$.

Step 1: Solve the subproblem (19) to evaluate $v(\gamma^s, u^s)$, and obtain the corresponding solution $\overline{x}^s$. Let $z^s \equiv v(\gamma^s, u^s)$, and if $s > N_2$, set $\tilde{x} \leftarrow \tilde{x} + \overline{x}^s$. Compute the subgradient g^s of $v(\cdot)$ at $\gamma = \gamma^s$ and $u = u^s$ having components $(g1_i^s, g2_k^s)$, where $g1_i^s = \left(\sum_{j=1}^{t_i}(j + d_i + \Delta_i)\overline{x}_{ij}^s\right)$, $i = 1, ..., N$, and $g2_k^s = \left(\sum_{(i,j)\in J_k} \overline{x}_{ij}^s - 1\right)$, $k = 1, ..., T$. If $z^s > \hat{z}$, then set $\hat{z} = z^s$ and store (γ^s, u^s) as the incumbent dual solution. If $s > N_2$ in addition to the previous condition $z^s > \hat{z}$, determine the current primal estimate $\hat{x} = \tilde{x}/(s - N_2)$, and compute an updated upper bound $\overline{z}$ based on the heuristic described in Section 5.5.4 below, provided that N_3 iterations have been executed since last invoking this heuristic.

Step 2: Compute a direction d^s having components $d1_i^s$ for $i = 1, ..., N$ corresponding to the γ variables and $d2_k^s$ for $k = 1, ..., T$ corresponding to the u variables, and a step size λ^s for updating the Lagrangian multipliers γ and u. Appropriate choices for d^s and λ^s are given in Sections 5.5.2 and 5.5.3, respectively. Set $u^{s+1} = P_{u\geq 0}[u^s + \lambda^s d1^s]$, where $P_{u\geq 0}[\cdot]$ projects $[\cdot]$ onto $\{u\colon\ u \geq 0\}$, and $\gamma^{s+1} = P_{\gamma\geq 0, e\gamma=1}[\gamma^s + \lambda^s d2^s]$, where $P_{\gamma\geq 0, e\gamma=1}[\cdot]$ projects $[\cdot]$ onto $\{\gamma\colon\ \gamma \geq 0,\ e\gamma = 1\}$. Note that the latter projection operation may be done via a variable dimension projection method, as detailed in [1]. Increment s by 1. If $s > N_1$, $(\overline{z} - \hat{z})/\overline{z} \leq \epsilon_1$, or $||g^s|| \leq \epsilon_2$, then terminate. Otherwise, return to Step 1.

5.5.1 Initial Upper Bounding Heuristic

To provide a quick upper bound to the problem, we execute an adaptation of the heuristic of Farina and Neri [3] (we will later compare our final upper bound to this initial value). The original version of this heuristic, summarized below, assumes that $\tau_i = 1$ and $\Delta_i = 0$ for all

$i = 1, ..., N$. A modification of this algorithm to accommodate more general values of τ and Δ is straightforward.

Farina and Neri Heuristic (FNH) for Problem P2

Initialization. Sort all the targets in nonincreasing order of d_i, and assume that these are re-indexed as $i = 1, ..., N$. Let $j = 2$, $cnt_R = cnt_L = 1$, $T = \sum_{i=1}^{N} d_i + N$, $T_0 = 0$, and $\mathcal{N} = N$. Schedule the transmission of target 1 in slot 1, and its reception in slot $d_1 + 1$.

Step 1: Schedule reception of targets after previous target receptions. If $d_j \leq cnt_R$, proceed to Step 2. Otherwise, schedule the transmission of target j in slot $d_1 + j - d_j + T_0$, and its reception in slot $d_1 + j + T_0$. If $j = \mathcal{N}$, terminate; else, increment j and cnt_R by one, and repeat Step 1.

Step 2: Schedule reception of targets before previous target receptions. Let $J = j$, and let $\hat{d} = d_{J-1} - cnt_R$, *i.e.*, $\hat{d}$ is the number of slots between the last transmitted pulse and the first received pulse. Consider each target $j = J, ..., \mathcal{N}$ in turn. If $d_j < \hat{d}$, then schedule the transmission of target j in slot $d_1 - cnt_L - d_j + 1 + T_0$, and its reception in slot $d_1 - cnt_L + 1 + T_0$, increment cnt_L by one, and set $\hat{d} = d_j - 1$. Repeat for the next j, until $j = \mathcal{N}$.

Step 3: Interleave targets in any available slots. For each unscheduled target $j \in \{J, ...\mathcal{N}\}$ in turn, find the first unoccupied slot $k \in \{T_0 + 2, ..., T_0 + d_1\}$ such that slots k and $k + d_j$ are unoccupied. If such a slot exists, schedule the transmission of target j in slot k, and its reception in slot $k + d_j$.

Step 4: Check for termination. Terminate if all targets have been scheduled. Otherwise, let $T_0 \leftarrow T_0 + d_1 + cnt_R$. Let $\mathcal{N}$ be the number of unscheduled targets, and re-index these targets as $i = 1, ..., \mathcal{N}$ as before. Schedule the transmission of target 1 in slot T_0+1, and its reception in slot $T_0 + d_1 + 1$. If $\mathcal{N} \geq 2$, return to Step 1, with $j = 2$ and $cnt_R = cnt_L = 1$; otherwise, terminate.

5.5.2 Method for Selecting the Direction d^s

We again utilize the ADS scheme for updating the direction, setting $d^s = g^s$ at $s = 1$, and for $s \geq 2$, setting

$$d^s = g^s + \frac{||g^s||}{||d^{s-1}||} d^{s-1}. \tag{20}$$

5.5.3 Method for Selecting the Step Size λ^s

Similar to HLDP1, we set the initial step size for this procedure according to

$$\lambda^1 = \overline{\beta} \frac{(\overline{z} - z^1)}{||d^1||^2}, \tag{21}$$

and we again retain this initial step length until we reach some reset criterion. In lieu of resetting the dual to its incumbent value and recomputing the step length after some N_{1b} iterations as in HLDP1, we instead track the number of consecutive iterations that fail to improve the lower bound. When this number of iterations reaches some parameter N_4, we reset the incumbent dual value and set the length of the new step size to half the length of the previous step size.

5.5.4 Upper Bounding Heuristic at Step 1 of HLDP2

Given the current primal solution estimate $\hat{x}$, let $y_i \in \arg\max_{j=1,...,t_i}\{\hat{x}_{ij}\}$, $i = 1, ..., N$ and consider the solution given by pulsing each target i at time slot y_i. Similar to the heuristic of Section 4.4.4, define f_k, $k = 1, ..., T$, to be the number of targets occupying slot k in excess of one, and let $F = \sum_{k=1}^{T} f_k$. If no infeasibility exists (i.e., $F = 0$), then terminate the heuristic. Otherwise, define Δ to be some positive integer, and set $\alpha_i = 1$ for $i = 1, ..., N$ and proceed to Step 1.

Step 1: For each target $i = 1, ..., N$, examine the possible beginning pulsing times for i from α_i to t_i, excluding y_i, that would yield a decrease in F. Let M be the set of all such possible moves (i, y_i, y_i^l), where y_i^l would be the new value of y_i under move l. If $M = \emptyset$, go to Step 2. Otherwise, go to Step 3.

Step 2: Let $T \leftarrow T + \Delta$, and let $\alpha_i = t_i + 1$ and $t_i \leftarrow t_i + \Delta$ for $i = 1, ..., N$. Return to Step 1.

Step 3: Find the move l^* in M that maximizes the decrease in F. Set $y_i = y_i^{l^*}$ and go to Step 4.

Step 4: Revise f and F based on the modified current solution. If $F = 0$, terminate; otherwise, return to Step 1.

6 Computational Results: Applications to Some Attack Scenarios

In this section, we investigate three experiments (denoted A, B, and C below) pertaining to the following two particular attack scenarios. The test cases generated by these experiments represent realistic data sets that typically arise in practice, and help provide insights into the relative performance of the various methodologies based on certain characteristics of the scenarios, as identified in the sequel.

1. Conventional massive air attack scenario, with distances ranging from 150-550 km, $N =$ 10-40 targets, pulses 2-10 μ-seconds, and target speeds of 180-1200 m/sec. Uncertainty in tracking distance may vary between a few hundred meters for nearby targets to 5000 meters for recently detected distant targets. (Experiments A and B pertain to this scenario.)

2. Ballistic attack scenario with 2-3 missiles, each dispatching 5-10 heads, decoys, and chaffs ($N =$ 10-30), with distances ranging from 50-150 km, speeds in the range of 3000-9000 m/s, and transmit pulse periods between 0.2 μ-seconds and 2.0 μ-seconds. Uncertainty is of the order of one or two basic pulse durations corresponding to a target position uncertainty between a few hundred meters at early detection to a few meters before target engagement. (Experiment C corresponds to this scenario.)

In the experiments described below, we define a performance measure for each of the two problems. For Problem 1, we simply report the weighted number of targets pulsed within the designated time limit. For Problem 2, in addition to the makespan required to pulse all the targets, we define another standard performance measure known as the *pulse packing*

efficiency that is given by

$$\eta = \frac{\text{Total time to track all targets sequentially}}{\text{Minimum time to interleave all targets}} = \frac{\sum_{i=1}^{N}(1 + d_i + \Delta_i)}{T_{min}}, \tag{22}$$

where T_{min} is defined as the optimal value for Problem P2. The following three experiments are performed using data derived from the foregoing two scenarios. All computations were conducted on a SUN Ultra-10 Workstation running Solaris 8, having 256 MB RAM. Also, for runs requiring the solution of linear/integer programs, we have utilized CPLEX 6.0.

6.1 Experiment A

We consider here the tracking process of a multifunction phased-array radar in the context of Problem P1. The planar phased array antenna rotates mechanically at a scanning rate of 5 rpm. Consequently, it scans the azimuthal angles every 12 seconds. The vertical sectors are scanned by steering the beam electronically. During tracking, the beam can further be steered in the azimuthal direction with respect to the mechanical axis in order to track all targets in a solid sector centered around the mechanical axis of the antenna. In a massive air attack scenario, we assume that we have N = 10-40 targets located in the sector to be scanned for tracking purposes. The targets are uniformly distributed over a radial distance of between 150 and 450 km. The radar pulse duration can be 10, 20, 30, 40 or 50 μ-seconds. The uncertainty Δ_i equals 10, 20, 30, or 40 μ-seconds. The time allocated for tracking is assumed to be 30 milliseconds. The objective here is to maximize the benefit function (5a) by appropriately selecting the targets that are to be tracked during the available period. The risk associated with each target depends on the type of the target, its speed, and its location. For generating the test cases, the cost factors are assumed to be given by

$$C_i = \left(\frac{450}{D_i}\right)^2 + v_i, \tag{23}$$

where D_i is distance of the target in kilometers, and v_i accounts for the target type and velocity, and is taken to be randomly distributed between 0.0 and 10.0.

For Experiment A, we generated test instances by varying both the number of targets N (using values of 10, 20, 30, and 40), and the duration of the scenario T (using values of 160,

200, and 240 μ-seconds). For each of the 12 combinations of N and T, we generated three test problems. Table 2 displays the results of our computational tests, where each entry is an average over the three test problems for the particular combination of N and T. If any of the three problems for a test set for a given N and T are not solvable within a 30-minute time limit (particularly for the CPLEX runs made to benchmark the heuristics), then the table provides the number of problems solved. (In this case, the average solution quality and computational results would be misleading, and are omitted.) For these problems, we solved P1 and P1$'$ (see Remark 2) to optimality, 5 percent of optimality, and 10 percent of optimality. We also display in Table 2 the lower bounding (feasible) solution value obtained via the heuristic developed in Section 4.4.1. These results demonstrate that P1 is an easier model to solve directly by a branch-and-bound solver than is P1$'$. We note that the heuristic solution provides a strong lower bound to the problem, but one that can be robustly improved upon by solving the problem to within 10 percent of optimality (although yet at up to two-orders of magnitude increase in computational effort).

The best option for solving this class of problems appears to be the use of HLDP1 of Section 4. (Parameter values used are $N_1 = 300$, $N_{1b} = 100$, $N_2 = 20$, $N_3 = 15$, $\overline{\beta} = 0.05$, and $\epsilon_1 = \epsilon_2 = 1$ x 10^{-3}.) Table 3 illustrates the computational results from testing this algorithm on the Experiment A test suite. The Lagrangian method identified very tight approximations to the linear programming solution in terms of the dual upper bounding objective function reported, in a considerably shorter time than that required by CPLEX. As for the lower bounds (feasible solution values) obtained from the algorithm, they are noticeably tighter than those generated by the initial heuristic. Overall, HLDP1 provided solutions within about 7% of optimality in less than approximately 4 cpu seconds, making it viable for practical scenarios.

6.2 Experiment B

In this experiment, we compare the effectiveness of our procedures versus the heuristic of Farina and Neri [3] (which is popularly used in practice) in the context of Problem P2. To mirror the analysis in [3], we designed Experiment B as follows. Each target was taken to have an equal pulse duration of $\tau = \lceil \frac{\lambda}{\delta} \rceil$ slots, where λ is the actual pulse duration, and δ is

the transmitter duty cycle. We took λ to be in the range of 1 to 10 μ-seconds and δ to be 0.2 or 0.3 μ-seconds. The targets were randomly positioned in a range of 2-50 km from the radar. Since the values of τ are small compared to the d-values, we normalized the resolution of the time slots such that $\tau_i = 1$ for all i. The received echo pulse durations were also taken to be equal to 1. Furthermore, since the uncertainty measure is not considered in [3], we let $\Delta_i = 0$ for all targets i. The packing ratio for this experiment is thus given by

$$\eta = \frac{\sum_{i=1}^{N}(1+d_i)}{T_{min}}, \tag{24}$$

where we have taken the unit time slot duration T_s to be equal to τ.

For Experiment B, we generated 5 sets of test problems containing five problems each, where set t contains $5(t+1)$ targets for $t = 1, ..., 5$. Table 4 demonstrates the solutions obtained by executing FNH, and by solving models P2 and P2$'$ to optimality, within 5 percent of optimality, and within 10 percent of optimality. We also record the amount of computational time required to solve models P2 and P2$'$. (Algorithm FNH executes very quickly for all problem instances.) We observe that although FNH is fast, it does not provide good quality solutions to these problems. Model P2$'$ is solvable to within 10 percent of optimality in less than about 5 cpu seconds for all problem instances, and requires substantially lesser computational effort than P2 (for the same optimality tolerance) on almost every problem instance. Furthermore, the solution yielded by P2$'$ matches the optimal solution provided by P2 on 17 out of the 19 cases on which both methods were able to solve problems to optimality. Table 5 demonstrates the increase in algorithmic efficiency that might be garnered by including the dual-based term in Problem P2$''$. Observe that by using a 5% optimality tolerance (which is achievable within about 1.4 cpu seconds for all problem instances), this problem solves exactly 19 of the 23 cases in which the optimal solution is known. In fact, P2$''$ consistently converges to a near optimal (within 0.3% of optimality) solution for every test problem, producing excellent results. The results of this experiment also suggest an efficient strategy that could be applied to other more general scheduling problems that involve the minimization of makespan (i.e., solve a problem of type P2$''$ in lieu of P2). In contrast with these results, the Farina-Neri Heuristic performs poorly with respect to solution quality as the value of N grows. For $N = 10$, the gap between the

Farina-Neri heuristic solution and the optimal solution is on average 18.7%. But as N grows to 15, 20, and 25, the average gap increases to 20.7%, 22.6%, and 26.7%, respectively. By examining the best known solutions for $N = 30$, we determine that the average gap between the Farina-Neri Heuristic and the optimal solution is at least 29.3%.

The results from using the Lagrangian dual approach are illustrated in Table 6. (Parameter values used are $N_1 = 200$, $N_2 = 1$, $N_3 = 15$, $N_4 = 15$, $\overline{\beta} = 0.05$, and $\epsilon_1 = \epsilon_2 = 1$ x 10^{-3}.) We were again able to very quickly find tight lower bounds to the linear relaxation of the problem within about 0.9 seconds of computational time, often affording a significant advantage over the CPLEX linear programming solver (this advantage will be more dramatically underscored in the next experiment). Observe that since all d_i and Δ_i for $i = 1, ..., N$ are integer-valued, we round up the lower bounds in our optimality gap calculations. However, the upper bounds provided by the algorithm do not compare well with those given by P2$''$, although still substantially better than the Farina-Neri heuristic. Overall, we therefore recommend the use of P2$''$ when solving problems of these dimensions. A comparison of the pulse packing efficiency measure η, as given by (22), for the Farina-Neri Heuristic versus the P2$''$ formulation is shown in Table 7, and further underscores the substantial improvements that our methodology provides.

6.3 Experiment C

In this experiment, we study the performance of the proposed technique under Scenario 2 in the context of Problem P2, because under such an attack, it is important to minimize the time required to track all the targets. The proposed solution is evaluated in terms of its packing efficiency. The target distances are assumed to follow a uniform distribution between 50 to 150 km. For a unit time slot of 0.1 μ-seconds, d_i is taken to vary from 3333 to 10000 time slots (333 to 1000 μ-seconds). The transmit pulse duration is uniformly distributed between 2-10 time slots (0.2 to 1.0 μ-seconds). Furthermore, the uncertainty is assumed to be $\Delta_i = 3\tau_i$, with a minimum receiver on-period duration of 2 μ-seconds. Similar to Experiment B, we generated 5 sets of test problems containing five problems each, where set t contains $5(t + 1)$ targets for $t = 1, ..., 5$.

For problems having such a large number of time slots as in this experiment, standard linear-programming based algorithms require prohibitive computational effort. Using the CPLEX 6.0 simplex solver, the average amount of computational time required to solve the linear relaxation of problems having 10 targets is about 1:15 minutes, while the average time required to solve the linear relaxation of problems having 30 targets is about 7:40 hours. These difficulties persist in models P2′ and P2″ as well. However, Table 8 demonstrates the ability of HLDP2 to both efficiently and accurately estimate the linear relaxation solution, and to obtain near-optimal solutions within 1-5 minutes of computational effort. For this set of problems, some parameters of the Lagrangian dual optimization scheme were altered to allow the algorithm to produce good answers in a short amount of time. We set the total number of iterations $N_1 = 50$, the reset counter $N_4 = 10$, and the initial step length parameter $\beta = 0.005$. Also, we only ran the upper bounding heuristic given in Section 5.5.4 once, at the end of the overall procedure. Yet, within this limited framework, we are still able to obtain good quality results for this class of problems. A comparison of the pulse packing efficiency afforded by HLDP2 and the Farina-Neri heuristic is given in Table 9. While HLDP2 produces improved solutions (relative increases in η of about 4% for problems with $N = 10$, and up to about 13% for problems with $N = 30$) as compared with the Farina-Neri heuristic as before, the solution effort required by the latter is considerably less (under 1 cpu second). Hence, for this class of problem instances where T is relatively large (due to large d_i/T_i ratios), we recommend investigating for future research a direct refinement of the Farina-Neri heuristic, rather than adopting any mathematical programming based approach.

7 Conclusions

Solution techniques for radar pulse interleaving problems have heretofore been confined to greedy heuristics that run quickly, but provide solutions that are often significantly worse than optimal. This paper presents insights into using new models and approaches for solving these problems based on contemporary mathematical programming techniques that provide tight approximations to the optimal solution, while not exceeding resource limitations (especially time) imposed by practical considerations. The potential of the proposed methods is demonstrated via three experiments pertaining to certain real scenarios. For Experiment

A, the proposed Lagrangian relaxation approach typically produced solutions within 7% of optimality in less than about 4 cpu seconds. Although the Lagrangian methodology also performed well on the test problems for Experiment B, we found that we may solve an altered integer programming formulation to within a very small (0.3%) optimality gap to quickly obtain solutions to the problem (within 0.9 cpu seconds). This methodology also suggests an efficient strategy that could be used in more general scheduling contexts involving the minimization of makespan. Furthermore, it might be of interest to explore theoretical properties of such approximation schemes for future research, in lieu of the empirical validation provided in the present study. For the test cases of Experiment C involving a more peculiar class of problems having relatively large pulse-return gaps and pulsing horizons, our results reveal that the proposed Lagrangian relaxation heuristic provides near optimal solutions considerably faster than exact methods, but yet, not as competitively as the Farina-Neri greedy heuristic. Hence for such problem scenarios, our results suggest that it might be more beneficial to refine the latter heuristic rather than to adopt more formal mathematical programming approaches. We also recommend this investigation for future research.

References

[1] M. Bazaraa, H.D. Sherali, and C.M. Shetty, Nonlinear Programming: Theory and Applications, 2nd edn, John Wiley & Sons, New York, NY, 1993.

[2] D. R. Carey, and W. Evans, The PATRIOT Radar in Tactical Air Defense, Microwave Journal 31 (1988), 325-332.

[3] A. Farina, and P. Neri, Multitarget Interleaved Tracking for Phased-array Radar, IEE Proceedings, Part F: Communications, Radar & Signal Processing 127 (1980), 312-318.

[4] M. Held, P. Wolfe, and H.P. Crowder, Validation of Subgradient Optimization, Mathematical Programming 6 (1974), 62-88.

[5] A. Izquierdo-Fuente, and J.R., Casar-Corredera, Optimal Radar Pulse Scheduling Using Neural Networks, 1994 IEEE International Conference on Neural Networks 7 (1994), 4588-4591.

[6] Y. Lee, and H.D. Sherali, Unrelated Machine Scheduling with Time-Window and Machine Downtime Constraints: An Application to a Naval Battle Group Problem, Annals of Operations Research, special issue on Applications to Combinatorial Optimization 50 (1994), 339-365.

[7] G.L. Nemhauser, and L. Wolsey, Integer and Combinatorial Optimization, John Wiley & Sons, Inc, New York, NY, 1988.

[8] M. Riezenman, Revising The Script After Patriot, Proceedings of the IEEE Spectrum 28 (1991), 49-52.

[9] H.D. Sherali, W.P. Adams, and P.J. Driscoll, Exploiting Special Structures in Constructing a Hierarchy of Relaxations for 0-1 Mixed Integer Problems, Operations Research 46 (1998), 396-405.

[10] H.D. Sherali, and G. Choi, Recovery of Primal Solutions When Using Subgradient Optimization Methods to Solve Lagrangian Duals of Linear Programs, Operations Research Letters 19 (1996), 105-113.

[11] H.D. Sherali, Y. Lee, and D. Boyer, Scheduling Target Illuminators in Naval Battle-Group Anti-Air Warfare, Naval Research Logistics 42 (1995), 737-755.

[12] H.D. Sherali, and O. Ulular, A Primal-Dual Conjugate Subgradient Algorithm for Specially Structured Linear and Convex Programming Problems, Applied Mathematics and Optimization 20 (1989), 193-224.

[13] M.I. Skolnik (ed), Radar Handbook, 2nd edn, McGraw Hill, New York, NY, 1990.

Model P1

Problem		Heuristic		Exact Optimality		5% of Optimality		10% of Optimality	
N	T	Ans	CPU	Ans	CPU	Ans	CPU	Ans	CPU
10	160	84.82	0.14	84.82	0.75	84.82	0.59	84.82	0.59
10	200	72.76	0.12	72.76	0.28	72.76	0.26	72.76	0.26
10	240	84.28	0.14	84.28	0.92	84.28	0.89	84.28	0.89
20	160	141.76	0.48	153.16	228.13	153.16	30.07	149.63	12.47
20	200	166.82	0.34	166.82	1.87	166.82	1.80	166.82	1.82
20	240	169.95	0.40	169.95	1.95	169.95	1.94	169.95	1.99
30	160	181.72	0.88	200.42	29.92	200.42	29.98	200.42	30.77
30	200	225.03	1.16	233.14	216.69	233.14	212.06	228.54	138.12
30	240	233.54	1.50	234.61	224.06	234.61	235.19	234.61	242.54
40	160	222.60	0.99	244.64	73.80	244.64	75.15	239.86	37.52
40	200	270.26	2.10	2 PS	2 PS	2 PS	2 PS	278.62	180.75
40	240	276.39	4.99	1 PS	1 PS	1 PS	1 PS	288.08	468.17

Model P1′

Problem		Exact Optimality		5% of Optimality		10% of Optimality	
N	T	Ans	CPU	Ans	CPU	Ans	CPU
10	160	84.82	1.50	84.82	1.38	84.82	1.23
10	200	72.76	0.66	72.76	0.58	72.76	0.53
10	240	84.28	2.27	84.28	2.15	84.28	1.91
20	160	0 PS	0 PS	2 PS	2 PS	146.98	46.50
20	200	166.82	79.42	164.36	16.44	164.36	14.83
20	240	169.95	12.91	169.95	14.38	169.95	12.84
30	160	0 PS	0 PS	199.82	547.74	197.15	89.00
30	200	0 PS	0 PS	2 PS	2 PS	227.81	757.90
30	240	1 PS	1 PS	2 PS	2 PS	2 PS	2 PS
40	160	0 PS	0 PS	2 PS	2 PS	237.17	183.78
40	200	0 PS	0 PS	2 PS	2 PS	284.32	704.96
40	240	0 PS	0 PS	0 PS	0 PS	0 PS	0 PS

Heuristic: Lower bounding heuristic described in Section 4.4.1

Ans: Best objective value yielded by the algorithm

CPU: Computational time, in seconds

PS: Number of problems solved within 30-minute time limit (if less than 3)

Table 2: Comparison of P1 and P1′ with the Initial Heuristic of Section 4.4.1.

Problem							
N	T	LB	UB	% Opt Gap	True % Opt Gap	% LP Gap	CPU
10	160	84.82	84.82	0	0	0	0.66
10	200	72.76	72.76	0	0	0	0.84
10	240	84.28	84.28	0	0	0	1.09
20	160	146.44	157.91	7.38	4.31	0.74	1.06
20	200	166.82	166.82	0	0	0	1.54
20	240	169.95	169.95	0	0	0	2.09
30	160	186.50	211.39	11.75	6.91	0.86	1.85
30	200	225.03	236.45	4.90	3.57	0	2.09
30	240	233.54	239.15	2.39	1.76	0	3.11
40	160	228.01	258.14	11.67	6.80	1.28	2.77
40	200	270.26	300.89	10.29	N/A	1.47	4.14
40	240	276.39	308.59	10.46	N/A	0.67	4.04

LB: Lower bound yielded by the algorithm

UB: Upper bound yielded by the algorithm

% Opt Gap: Gap between the upper and lower bound as a percentage of the upper bound

True % Opt Gap: Gap between the optimal solution (if known) and the lower bound as a percentage of the optimal solution.

% LP Gap: Lower bound gap as a percentage of the true LP solution

CPU: Computational time, in seconds

Table 3: Analysis of Lagrangian Relaxation Method on Experiment A Test Problems.

			Exact Optimality				5% of Optimality				10% of Optimality			
		FNH	P2		P2′		P2		P2′		P2		P2′	
Prob	N	Ans	Ans	CPU	Ans	CPU	Ans	CPU	Ans	CPU	Ans	CPU	Ans	CPU
1	10	76	66	0.59	66	0.59	66	0.89	66	0.55	66	0.7	66	0.67
2	10	63	54	2.06	54	0.52	56	1.96	54	0.48	58	1.32	54	0.55
3	10	61	51	0.6	51	0.54	51	0.61	51	0.51	51	0.7	51	0.56
4	10	68	58	0.93	58	0.49	59	0.61	58	0.47	59	0.62	58	0.47
5	10	50	40	16.69	41	0.8	42	2.78	41	0.55	44	1.62	41	0.58
6	15	77	64	5.33	64	0.75	66	3.47	64	0.75	66	3.88	64	0.7
7	15	79	66	5.51	66	1.2	68	4.2	67	0.7	71	0.96	67	0.86
8	15	78	65	4.06	65	0.67	66	2.72	65	0.65	66	2.69	65	0.64
9	15	78	66	5.67	66	0.75	66	5.5	66	0.72	66	5.61	66	0.76
10	15	69	55	4.12	55	0.91	55	4.19	55	0.81	58	4.05	55	0.81
11	20	81	67	12.5	67	1.57	68	6.38	67	1.1	72	5.77	67	1.03
12	20	82	66	34.83	66	1.22	69	21.26	67	1.01	69	21.54	67	1
13	20	79	65	49.07	65	1.42	67	44.56	65	1.05	67	44.21	65	1.05
14	20	67	55	1510.9	56	34.37	55	172.9	56	1.03	55	180	56	1.04
15	20	81	65	32.13	65	1.98	66	26.92	65	1.18	66	27.07	65	1
16	25	83	66	136.6	66	2.63	68	69.6	66	1.63	68	69.1	66	1.67
17	25	76	58	96.9	–	–	58	96.6	59	2.7	63	59.66	59	2.91
18	25	81	65	91.6	65	18.99	67	73.6	65	1.63	67	74.6	65	1.71
19	25	83	65	162.4	65	8.85	66	135.3	65	1.48	66	138.2	65	1.54
20	25	86	69	282.8	69	1.99	71	64.2	71	1.28	71	65.1	71	1.34
21	30	82	64	917.4	–	–	67	200.3	71	3.72	67	199.4	71	3.61
22	30	87	65	360.6	–	–	67	251.2	65	4.26	70	202.9	65	4.37
23	30	89	–	–	–	–	–	–	70	5.18	69	186.86	70	5.3
24	30	84	65	145.1	–	–	68	45.57	65	7.99	68	46.18	67	5.22
25	30	87	–	–	69	1799.2	–	–	69	2.04	–	–	69	2.03

FNH: Farina and Neri Heuristic

Ans: Best objective value yielded by the algorithm

CPU: Computational time, in seconds

–: Problem not solvable within a 30-minute time limit

Table 4: Comparison of P2 and P2′ with the Farina-Neri Heuristic.

			Exact Optimality		0.3% of Optimality		5% of Optimality	
		FNH	P2″		P2″		P2″	
Prob	N	Ans	Ans	CPU	Ans	CPU	Ans	CPU
1	10	76	66	0.27	66	0.27	66	0.14
2	10	63	54	0.15	54	0.16	54	0.15
3	10	61	51	0.16	51	0.16	51	0.15
4	10	68	58	0.16	58	0.17	58	0.15
5	10	50	40	0.2	40	0.18	40	0.18
6	15	77	64	0.19	64	0.19	64	0.19
7	15	79	66	0.23	66	0.19	66	0.2
8	15	78	65	0.19	65	0.18	65	0.19
9	15	78	66	0.2	66	0.2	66	0.21
10	15	69	55	0.22	55	0.21	55	0.2
11	20	81	67	0.26	67	0.26	68	0.26
12	20	82	66	0.29	66	0.25	66	0.26
13	20	79	65	0.29	65	0.25	65	0.25
14	20	67	55	0.42	55	0.3	55	0.3
15	20	81	65	0.56	65	0.27	65	0.27
16	25	83	66	0.7	66	0.35	66	0.36
17	25	76	–	–	58	1.03	58	0.6
18	25	81	65	2.68	66	0.37	66	0.39
19	25	83	65	1.28	65	0.36	65	0.78
20	25	86	69	0.37	69	0.41	69	0.34
21	30	82	–	–	66	1.84	66	0.59
22	30	87	–	–	66	1.4	66	1.01
23	30	89	–	–	69	10.9	73	1.42
24	30	84	–	–	65	7.58	65	1.34
25	30	87	69	26.14	69	0.57	69	0.58

FNH: Farina and Neri Heuristic

Ans: Best objective value yielded by the algorithm

CPU: Computational time, in seconds

–: Problem not solvable within a 30-minute time limit

Table 5: Comparison of P2″ with the Farina-Neri Heuristic.

Problem							
Prob	N	LB	UB	% Opt Gap	True % Opt Gap	% LP Gap	CPU
1	10	65.97	66	0	0	0.05	0.27
2	10	53.98	54	0	0	0.04	0.27
3	10	50.97	51	0	0	0.06	0.29
4	10	57.96	59	1.72	1.72	0.07	0.28
5	10	39.96	41	2.50	2.50	0.10	0.31
6	15	63.47	64	0	0	0.05	0.39
7	15	65.34	66	0	0	1.00	0.40
8	15	64.93	66	1.54	1.54	0.11	0.38
9	15	65.94	66	0	0	0.09	0.41
10	15	54.92	56	1.82	1.82	0.15	0.42
11	20	65.9	81	22.72	20.90	0.52	0.51
12	20	64.67	66	1.54	0	0.89	0.52
13	20	64.13	70	7.69	7.69	1.34	0.53
14	20	51.64	56	7.69	1.82	0.31	0.57
15	20	64.46	66	1.54	1.54	0.83	0.54
16	25	64.11	71	9.23	7.58	0.60	0.66
17	25	57.42	59	1.72	1.72	1.00	0.76
18	25	63.97	69	7.81	6.15	0.44	0.67
19	25	63.86	66	3.13	1.54	1.75	0.68
20	25	66.91	74	10.45	7.25	0.51	0.64
21	30	62.85	70	11.11	9.38	0.24	0.89
22	30	64.31	69	6.15	6.15	1.06	0.87
23	30	63.5	73	14.06	N/A	0.47	0.75
24	30	63.39	69	7.81	6.15	1.34	0.86
25	30	64.37	75	15.38	N/A	0.97	0.80

LB: Lower bound yielded by the algorithm

UB: Upper bound yielded by the algorithm

% Opt Gap: Gap between the upper and (rounded up) lower bound as a percentage of the lower bound

True % Opt Gap: Gap between the upper bound and the optimal solution (if known) as a percentage of the optimal solution.

% LP Gap: Upper bound gap as a percentage of the true LP solution

CPU: Computational time, in seconds

Table 6: Analysis of the Lagrangian Relaxation Method on Experiment B Test Problems.

Problem		Packing Efficiency η		True % Opt Gap	
Prob	N	FNH	P2″ (0.3%)	FNH	P2″ (0.3%)
1	10	4.36	5.02	15.15	0
2	10	5.92	6.91	16.67	0
3	10	5.52	6.61	19.61	0
4	10	5.18	6.07	17.24	0
5	10	6.24	7.80	25.00	0
6	15	7.21	8.67	20.31	0
7	15	6.95	8.32	19.70	0
8	15	7.76	9.31	20.00	0
9	15	6.53	7.71	18.18	0
10	15	7.41	9.29	25.45	0
11	20	8.16	9.87	20.90	0
12	20	9.07	11.27	24.24	0
13	20	7.95	9.66	21.54	0
14	20	9.63	11.73	21.82	0
15	20	8.95	11.15	24.62	0
16	25	10.95	13.77	25.76	0
17	25	9.72	12.74	31.03	0
18	25	10.30	12.64	24.62	1.54
19	25	10.31	13.17	27.69	0
20	25	11.48	14.30	24.64	0
21	30	12.32	15.30	28.13	3.13
22	30	11.95	15.76	33.85	1.54
23	30	13.65	17.61	N/A	N/A
24	30	10.93	14.12	29.23	0
25	30	13.05	16.45	N/A	N/A

Table 7: Comparison of Pulse Packing Efficiencies and Optimality Gaps for Experiment B Test Problems.

Problem						
Prob	N	LB	UB	% Opt Gap	% LP Gap	CPU
1	10	9097.96	9119	0.23	0.0055	50.87
2	10	8944.98	8945	0	0.0002	47.11
3	10	9949.99	9950	0	0.0001	80.22
4	10	8652.88	8692	0.45	0.0014	59.46
5	10	9752.85	9753	0	0.0015	62.37
6	15	9900.88	9901	0	0.0012	105.08
7	15	9484.86	9485	0	0.0015	89.97
8	15	9800.87	9801	0	0.0013	92.61
9	15	9974.89	9975	0	0.0011	142.36
10	15	8142.91	8143	0	0.0011	86.21
11	20	9754.12	9806	0.52	0.0090	155.86
12	20	9725.59	9726	0	0.0042	178.12
13	20	9649.88	9650	0	0.0012	192.11
14	20	9874.01	9875	0	0.0100	160.26
15	20	9670.49	9728	0.59	0.0053	171.71
16	25	9921.66	9922	0	0.0034	255.58
17	25	9636.7	9637	0	0.0031	218.96
18	25	9863.52	9864	0	0.0049	246.72
19	25	9963.1	9964	0	0.0090	320.65
20	25	9544.75	9545	0	0.0026	247.7
21	30	9705.7	9859	1.58	0.1226	323.3
22	30	9711.05	9775	0.65	0.0098	395.14
23	30	9863.19	9927	0.64	0.0082	303.99
24	30	9862.68	9964	1.02	0.0479	121.64
25	30	9617.56	9618	0	0.0046	175.04

LB: Lower bound yielded by the algorithm

UB: Upper bound yielded by the algorithm

% Opt Gap: Optimality gap as a percentage of the (rounded up) lower bound

% LP Gap: Upper bound gap as a percentage of the true LP solution

CPU: Computational time, in seconds

Table 8: Analysis of the Lagrangian Relaxation Method on Experiment C Test Problems.

Problem		Packing Efficiency η	
Prob	N	FNH	HLDP2
1	10	7.24	7.52
2	10	7.71	8.09
3	10	5.79	6.06
4	10	7.08	7.38
5	10	6.98	7.31
6	15	10.53	11.25
7	15	10.45	11.18
8	15	9.80	10.38
9	15	8.94	9.51
10	15	10.16	10.84
11	20	13.17	14.34
12	20	13.17	14.29
13	20	11.90	12.94
14	20	13.32	14.54
15	20	11.40	12.37
16	25	15.32	17.12
17	25	14.88	16.48
18	25	14.07	15.57
19	25	15.42	17.23
20	25	15.32	17.05
21	30	18.50	20.80
22	30	16.96	19.16
23	30	17.96	20.53
24	30	19.17	21.40
25	30	18.34	20.88

Table 9: Comparison of Pulse Packing Efficiencies for Experiment C Test Problems.