On Scheduling a Multifunction Radar

Émilie Winter[1,2], Philippe Baptiste[1]

[1] Lix, Cnrs, École Polytechnique, 91128 Palaiseau, France.
{emilie.winter, philippe.baptiste}@lix.polytechnique.fr
[2] Thales Systèmes Aéroportés, 2, av Gay-Lussac, 78851 Elancourt, France.
emilie.winter@fr.thalesgroup.com

Abstract :

Among several other tasks, the radar of a fighter has to search, track and identify potential targets. The waveforms used by the radar for each of these tasks are most often incompatible and hence, cannot be processed simultaneously. Moreover, these tasks are repeated several times in a cyclic fashion. Altogether, this defines a complex scheduling problem that impacts a lot on the quality of the radar's output. In this paper, we define a formal framework for this real time scheduling problem and we introduce several techniques to compute efficient schedules for the radar. Experimental results are provided.

Keywords : Scheduling, Local Search, Linear Programming, Radar.

1 Introduction

A radar is a system using radiowaves to detect the presence of objects in a given domain. It can also compute the range (distance) as well as the relative radial velocity of these objects. Airborne radars consist of a transmitter, a single antenna and a receiver. The transmitter generates radiowaves which are sent out in a narrow beam by the antenna in a specific direction. Objects located in the beam intercept this signal and scatter the energy in all directions. A portion of this energy is scattered back to the receiver of the radar listening to all potential echoes. See (8) and (14) for a detailed description of airborne radars.

Nowadays, most of the airborne radars have a mechanically steered antenna. With such antennas, the beam is perpendicular to the antenna which is pivoting so as to direct the beam. While working, the radar uses a "track-while-scan" technique: Search tasks are executed by sweeping across a search domain following a "wind-screen wiper" strategy. Dwells of other types (in particular tracking tasks) are played in passing.

This paper is focused on recent radars with Electronically Steered Antenna (ESA). An ESA is a planar array antenna made of many individual radiating elements. Unlike a mechanically steered antenna, an ESA lies in a fix position on the aircraft. The phase of the radiowaves is controlled electronically so that the radar beam lights up the desired direction.

One of the key advantages of ESA is that the beam is extremely agile. As it is not subjected to the mechanical inertia of the antenna, it can be moved instantaneously from one part of the space to another, even outside the search domain (see Figure 1). Moreover, the radar can switch instantaneously to an appropriate waveform.

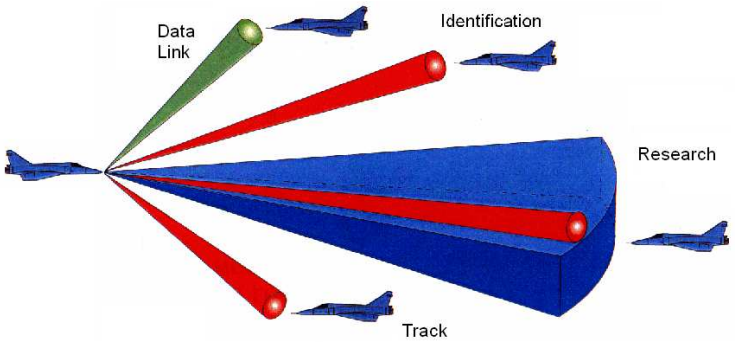

Figure 1: Advantages of ESA Radars

The following tasks have to be achieved by an airborne radar.

- **Research.** The radar is sweeping across a domain to detect the potential presence of targets inside.
- **Tracking.** The radar is closely monitoring the behavior of targets (initially detected at the Research stage).
- **Data Link.** The radar is used as a communication tool with other platforms.
- **Calibration.** The radar is performing cyclic calibration to ensure a high reliability level.

Research, Tracking, Data Link and Calibration require incompatible waveforms. So all corresponding tasks have to be *scheduled to ensure they do not overlap in time*. Moreover, Research, Tracking, Data Link and Calibration tasks have to be repeated in a more or less regular fashion. Depending on the task, the periodicity constraint can be extremely important or not. For instance, as it is absolutely forbidden to loose a tracked target, tracking dwells have to be repeated with high regularity. Likewise, data link dwells are played at regular intervals, but there, the regularity constraint is much higher. Finally, to ensure a surveillance of great quality, in any circumstances, the radar has to play at least a minimum amount of research dwells.

The aim of the study is to model the radar and to make it "more efficient" while meeting all constraints informally described above. We also want to take into account situations where the radar is overloaded and where some tasks have to be rejected by the system.

Barbaresco (2) describes a strongly related framework: Tasks are scheduled on a frame duration and are executed while the next schedule is computed. To schedule the tasks, Barbaresco associate deadlines to tasks and use a heuristic called EDF (Earliest Deadline First). If one of the most important tasks is completed after its deadline then some tasks are removed of the system to reduce the load and the scheduling procedure is run again. The empirical study led in (2) shows that simple scheduling heuristics often improve the behavior of the radar.

The scheduling of the radar tasks can also be seen as a timing problem in which the tasks are scheduled to minimize an Earliness-Tardiness criterion (13).

Another scheduling problem for radars is the problem of interleaving tasks corresponding to receiving and sending data (5; 12). We study a situation where we do not have interleaving and both the sending and the receiving tasks are modeled as a unique task.

Our objective is to introduce a *formal model* of the problem to be able to evaluate the quality of solutions (Section 2). We show in Section 3 that the problem is strongly NP-Hard

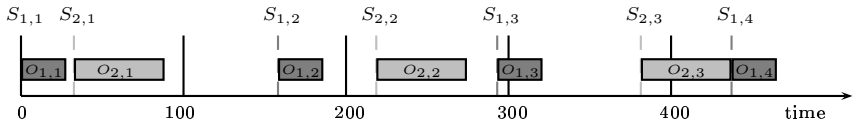

Figure 2: Regularity Constraints

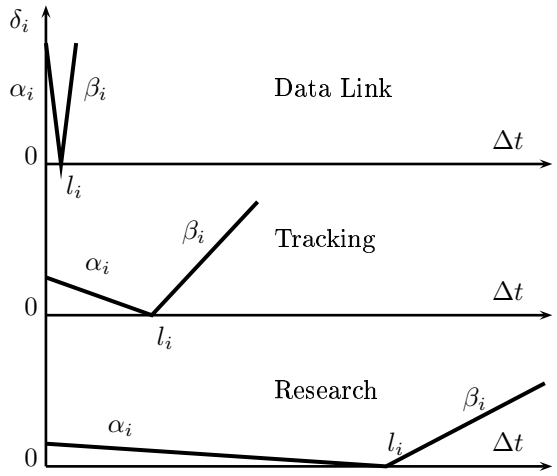

Figure 3: Cost Functions

and that a special case can be solved in polynomial time by Linear Programming. Then, we introduce a local search method and two fast real-time scheduling procedures (Section 4) that are experimentally compared in Section 4.3. Finally, we compute lower bounds to evaluate the quality of our heuristics (section 5).

2 Model

We define a "*job*" as a single radar function that must be repeated in a more or less strict manner. A job is either the track of a given target or a data link or the "research". As several dwells are required to "execute" most jobs, a job consists in a set of non-preemptive "operations" which must obey a kind of periodicity rule. As shown on figure 2, jobs are not necessarily executed with strict periodicity. The temporal gap between the two first operations of job 1 is greater than the gap between the operations 2 and 3. This schedule is feasible if performance constraints are ensured. The radar is seen as a single machine on which jobs have to processed.

To model the problem, we associate cost functions δ_i to the distance between starting times of consecutive operations of the same job i. They are V-Shaped, *i.e.*, $\delta_i(x) = \max(\alpha_i(l_i - x), \beta_i(x - l_i))$ where $l_i, \alpha_i \geq 0$ and $\beta_i \geq 0$ are respectively the ideal distance between two starting times and the penalty weights associated to a smaller (resp. larger) inter-distance.

For each kind of job i, we have defined, in collaboration with radar engineers, cost functions features that fit real life scenari. An example of what cost functions can be is described in Figure 3.

We are now ready to formally define the problem. Given n jobs 1, 2, ..., n. Each job i is made of $n(i)$ consecutive and identical operations $O_{i1}, ..., O_{in(i)}$ with processing time p_i. For

each job i, we have a penalty function δ_i associated to the distance between starting times of consecutive operations. We are also given an integer $H \geq \sum_i n(i)p_i$ that represents the horizon of the schedule (*i.e.*, all operations have to be processed between 0 and H).

A set of starting times S_{ij} defines a feasible schedule if and only if (1) operations start after or at 0 and are not completed later than H and (2) operations do not overlap, *i.e.*, for any pair of operations $O_{ij}, O_{i'j'}$, $S_{ij} + p_i \leq S_{i'j'}$ or $S_{i'j'} + p_{i'} \leq S_{ij}$. The cost associated to the schedule is exactly the sum over all jobs i of $\sum_{u=1}^{n(i)-1} \delta_i(S_{iu+1} - S_{iu})$.

Note that, in practice, the problem is solved continuously and the schedule followed in the past interacts with the schedule under construction. Stated another way, many jobs have been started a long time ago. This feature does not change the combinatorial structure of the problem and to keep things simple, we omit all details about it.

3 Complexity and Related Results

We first prove that the problem is NP-Hard in the strong sense (6) we then show that the variant of the problem, in which the order between all operations is known, can be solved in polynomial time by linear programming. This latest property is at the root of the scheduling heuristic proposed in 4.2

3.1 NP-Hardness

The problem obviously belongs to NP and we prove it is NP-Hard in the strong sense. We first recall the well-known 3-Partition problem.

- Given a sequence A of positive integer values $a_1, ..., a_{3s}$ and a positive integer value T such that $\frac{T}{4} < a_i < \frac{T}{3}$ for all $1 \leq i \leq 3s$, and such that $\sum_{i=1}^{3s} a_i = sT$.
- Can A be divided into s disjoint subsets $B_1, ..., B_s$ such that: $\forall j, \sum_{a_i \in B_j} a_i = T$?

Now consider an instance of 3-Partition and build an instance of the decision variant of our scheduling problem with threshold equal to 0 as follows: $H = sT + s + 1$ and create $3s$ "regular" jobs i with one operation ($n(i) = 1$), processing time p_i and $\forall t, \delta_i(t) = 0$. We also have one "blocking" job $3s + 1$ with $p_{3s+1} = 1$, $n(i) = s + 1$ and $\delta_{3s+1}(t) = |t - (T + 1)|$.

Proposition 1. *There is a solution for the 3-Partition instance if and only if there is a solution for the scheduling instance.*

Proof. Consider a solution of the scheduling instance. As the threshold is 0 the distance between the starting times of consecutive operations of the blocking job is exactly $T+1$. Due to our choice of the time horizon, the starting time of the uth blocking operation is exactly $(u - 1) \times (T + 1)$. Let us define B_u as the set of regular jobs scheduled between the u^{th} and the $u + 1$th operation of the blocking job. It is easy to see that there is no idle time and thus, the sum of the processing times of the jobs in B_u is exactly T.
Now assume we have a solution to the 3-Partition instance. Let then $B_1, ..., B_s$ denote the disjoint subsets such that: $\forall j, \sum_{a_i \in B_j} a_i = T$. We build a feasible schedule as follows : The u^{th} operation of the blocking job starts at $(u - 1) \times (T + 1)$ and we schedule the jobs corresponding to B_u in any order between the u^{th} and the $(u + 1)^{th}$ operation of the blocking job. It is easy to check that the schedule is feasible and that its cost is 0. □

3.2 A Polynomial Time Algorithm for a Fixed Sequence

Consider the variant of the problem in which the order of all operations is known. Now the question is to compute starting times (meeting the order mentioned above) that minimize the total cost. This problem can be solved in polynomial time because cost functions are V-shaped (in general this problem is strongly NP-Hard). We show that the problem can be modeled as a Linear Program. A Linear Program (LP) is a problem that can be expressed as follows $\min\{cx : Ax = b, x \geq 0\}$ where x is the vector of variables A is a matrix, and c, b are vectors. Several methods, including the "Simplex" and the "Interior-Point" methods, allow to solve large linear programs within a reasonable amount of CPU time. We refer to (9) and (11) for a complete introduction to LP.

From now on, assume that for any job i and any operation j, $pred(i,j)$ is the couple of indices (i',j') that corresponds to the operation $O_{i'j'}$ preceding immediately O_{ij} in the sequence. Assume that $pred(i,j) = (0,0)$ if O_{ij} is the first operation in the sequence.

We introduce a simple Linear Program (LP) in which S_{ij} denotes the starting time of O_{ij}. W_{ij} is the cost associated to the distance between O_{ij} and O_{ij-1}.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n(i)} W_{ij}$$

$$(LP) \begin{cases} S_{i'j'} + p_{i'} \leq S_{ij} & 1 \leq i \leq n, 1 \leq j \leq n(i) \\ & (i',j') = pred(i,j) \neq (0,0) \\ W_{ij} \geq \alpha_i(l_i - S_{ij} + S_{ij-1}) & 1 \leq i \leq n, 2 \leq j \leq n(i) \\ W_{ij} \geq \beta_i(S_{ij} - S_{ij-1} - l_i) & 1 \leq i \leq n, 2 \leq j \leq n(i) \\ 0 \leq S_{ij} \leq H - p_i & 1 \leq i \leq n, 1 \leq j \leq n(i) \\ 0 \leq W_{ij} & 1 \leq i \leq n, 2 \leq j \leq n(i) \end{cases}$$

Proposition 2. *The above Linear Program (LP) computes an optimal set of starting times.*

Proof. Note that because of the constraints $S_{i'j'} + p_{i'} \leq S_{ij}$, any solution of the LP defines a feasible schedule that meets the order of the initial sequence. Moreover, the cost of a schedule is exactly

$$\begin{aligned} & \sum_{i=1}^{n} \sum_{j=2}^{n(i)} \delta_i(S_{ij} - S_{ij-1}) \\ = & \sum_{i=1}^{n} \sum_{j=2}^{n(i)} \max(\alpha_i(l_i - S_{ij} + S_{ij-1}), \beta_i(S_{ij} - S_{ij-1} - l_i)) \\ \leq & \sum_{i=1}^{n} \sum_{j=2}^{n(i)} W_{ij} \end{aligned}$$

Note that if the inequality is strict then there is at least one variable W_{ij} such that $W_{ij} > \alpha_i(l_i - S_{ij} + S_{ij-1})$ and $W_{ij} > \beta_i(S_{ij} - S_{ij-1} - l_i)$. So we can reduce W_{ij} of some small value and decrease the objective function. This contradicts the optimality of the solution. □

4 Heuristics

The schedule of the radar has to be built within some fractions of a second. We then have to derive fast real time heuristics to solve the problem. In the following we describe two simple

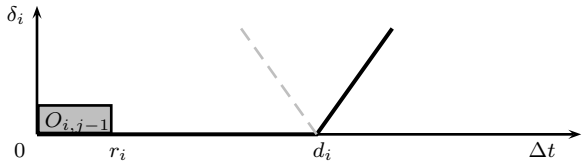

Figure 4: Definition of Release Dates and Deadlines for EDD1

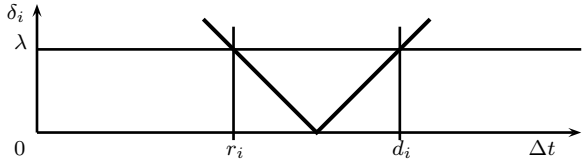

Figure 5: Definition of Release Dates and Deadlines for EDD2

heuristics based on estimated deadlines. We then describe a local search method. It is a bit more costly in terms of CPU time but it builds good schedules which serve as references for our fast real time heuristics.

4.1 Basic Heuristics

Our heuristics build schedules iteratively from left to right. Let t denote the completion time of the last scheduled operation. Some operations have already been scheduled (before t) while some other operations are waiting to be scheduled. As before, S_{ij} denotes the starting time of O_{ij}.

To determine which operation is to be scheduled next the EDD1 heuristic computes, for each unscheduled operation O_{ij}, an estimated deadline that equals $S_{ij-1}+l_i$ if $j > 1$ and l_i otherwise. The next operation is the one with minimal estimated deadline.

The second heuristic EDD2 is slightly more complex. We compute a time window for each unscheduled operation O_{ij}. This window is computed according to a fixed threshold value $\lambda > 0$ and to the following rules.

- If O_{ij} is the first operation of its job, *i.e.*, $j = 0$, the release date (respectively deadline) of O_{ij} is defined as the smallest (resp. largest) time point x such that $\delta_i(x) \leq \lambda$.
- If O_{ij} is not the first operation of its job, the release date (respectively deadline) of O_{ij} is defined as the smallest (resp. largest) time point x such that $\delta_i(x - S_{ij-1}) \leq \lambda$.

EDD2 schedules at time t the operation with minimal deadline among operations that have a release date greater than or equal to t.

4.2 Local Search Method

Local search is a well-known methodology used for obtaining good solutions of NP-hard combinatorial optimization problems. A local search algorithm starts with a feasible solution x and tries to find a better solution y in the neighborhood of x. If such a solution y is found then the current solution becomes y and the algorithm iterates. otherwise, the algorithm ends with the locally optimal solution x. We refer to (1) for an overview of local search and to (3; 4) for a detailed study of local search methods applied to scheduling problems. A key feature of efficient Local Search techniques lies in the definition of the neighborhood.

Recall that for our problem when the sequence of all operations is known, the problem can be solved in polynomial time by linear programming (Section 3). Hence, a solution can be defined by the sequence of operations. Our local search is based on this encoding of solutions that allows us to rely on well-known neighborhood structures. Moreover, this encoding is much more compact than the one based on finding directly the exact starting times. We start with some random sequence that we improve step by step until iteratively try to improve (see Figure 6). Neighbors are obtained by either swapping two randomly chosen operations (SWR) or by swapping consecutive operations (SWC), or by inserting a randomly chosen operation at some other place in the sequence (IR). The "moves" SWR and IR can lead to sequence of operations that do not meet the natural order of operations within one job. For instance O_{ij} can be switched (or inserted) after O_{ij+1} (or before O_{ij-1}) As all jobs' operations have the same processing times and the same cost functions, the sequence is "repaired" by renumbering operations. We have also designed two other variants of the moves SWRL and IRL (where "L" stands for Limited) in which sequences that do not meet the natural order of operations are not considered.

A move is performed if the schedule corresponding to the new sequence improves the best known total cost. This step requires the computation of a linear program and, as shown by initial experiments, this turns to be very costly in terms of CPU time. To improve the behavior of the algorithm, we have designed a set of dominance rules that allow us to discard some of the moves that do not improve the solution.

Let us denote O_{ij} and O_{kl} the two operations chosen for a move and assume that O_{ij} is before O_{kl} in the sequence. Our dominance rules consist in evaluating, by using the current and the new sequence, the best possible cost variation on the modified part of the sequence, that is between the end of $pred\{i,j\}$ and the begining of the operation following O_{kl} in the sequence. If the new sequence may improve the cost, we compute the schedule.

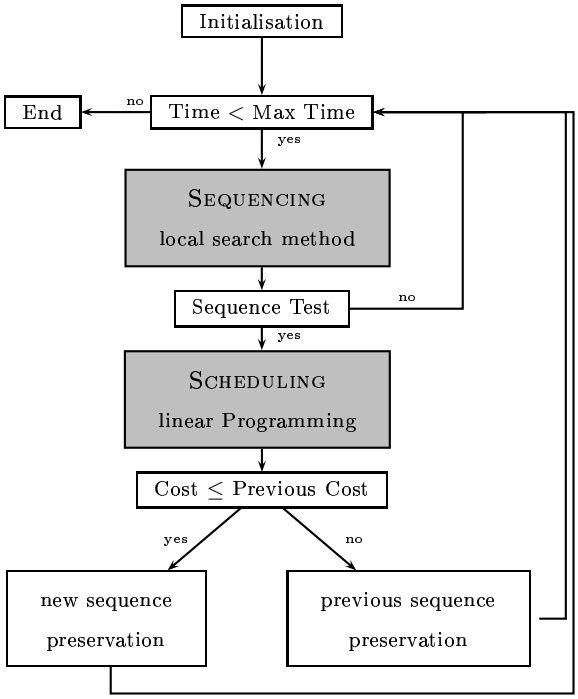

Figure 6: An Overview of Local Search

4.3 Experiments

Together with Thales engineering department, we have generated 27 random instances that represent real life situations. The total number of operations varies from 44 to 96, the load of the radar varies from 45% to 100 % and the slopes of the cost functions α and β have been chosen to model different kind of situations. All experiments have made run on DELL Latitude D600 laptop running Linux. We have used GNU Linear Programming Kit (7) as a LP solver.

We first report computational results of the 5 local search methods SWR, SWC, IR, SWRL, IRL over the 27 instances. For each instance, we denote by ub_t^X the value of the objective function after t seconds of CPU time for the local search X (either SWR, SWC, IR, SWRL or IRL). We define $*ub_\infty$ as the best solution found over all methods. To report on the average behavior of the algorithms, we normalize all objective values. Figure 7 reports the average normalized objective function

$$\frac{ub_t^X - ub_\infty}{ub_0^X - ub_\infty^X}$$

of the 5 methods over time.

This clearly shows that SWRL, IRL and SWR outperform all other methods. Moreover, the the local search algorithms converge fast.

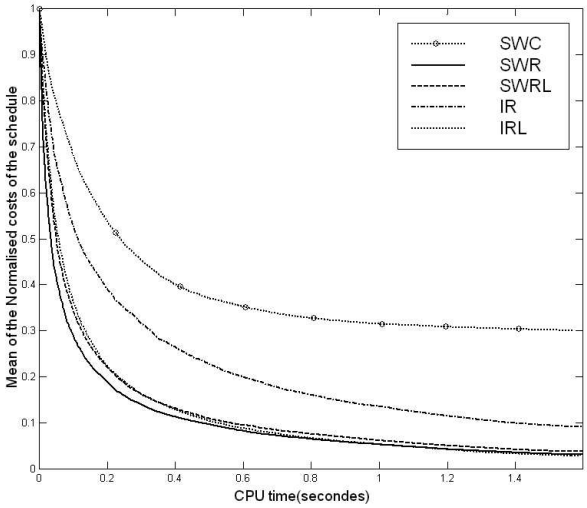

Figure 7: Average Normalized Objective Values

Recall that EDD2 can be run for several values of λ. We look for the best possible λ value by a partial enumeration of this parameter. Compared to the local search algorithms, the EDD heuristics do not behave well at all. On the average, the ratio between EDD1 and the best solution obtained by the 5 local search methods, is more than 17 while compared to EDD2 it is greater than 13. Compared with the worse solution obtained by the 5 local search methods, EDD1 and EDD2 are within a factor of 3.

5 Lower Bounds

We compute lower bounds to evaluate the quality of our heuristics. They are obtained by relaxing integer linear programs.

5.1 Two Linear Programming Methods

We can compute the total minimum cost of the whole problem (sequencing and scheduling) by Mixed Integer Linear Programming. Two methods are proposed below.

A) *Method LB Disjunctive :* We define $X_{(i'j')(ij)}$ a binary variable which represents precedences between tasks : $X_{(i'j')(ij)}$ equals 1 if $O_{i'j'}$ precedes O_{ij}, 0 otherwise. The first two constraints are due to the cost functions (see 3.2). In the third one, we use a "big" artificial constant M to say that tasks cannot overlap in time. Next come the triangular inequalities which fix the sequence of the operations.

$$\min \sum_{i=1}^{n} \sum_{j=2}^{n(i)} W_{ij}$$

$$\begin{cases} W_{ij} \geq \alpha_i(l_i - S_{ij} + S_{ij-1}) & 1 \leq i \leq n, 2 \leq j \leq n(i) \\ W_{ij} \geq \beta_i(S_{ij} - S_{ij-1} - l_i) & 1 \leq i \leq n, 2 \leq j \leq n(i) \\ S_{i'j'} + p_{i'} * X_{(i'j')(ij)} - M(1 - X_{(i'j')(ij)}) \leq S_{ij} & 1 \leq i \leq n, 1 \leq j \leq n(i) \\ X_{(i''j'')(ij)} \geq X_{(i''j'')(i'j')} + X_{(i'j')(ij)} - 1 & 1 \leq i \leq n, 2 \leq j \leq n(i) \\ 0 \leq S_{ij} \leq H - p_i & 1 \leq i \leq n, 1 \leq j \leq n(i) \\ 0 \leq W_{ij} & 1 \leq i \leq n, 2 \leq j \leq n(i) \end{cases}$$

B) *Method LB Time Indexed :* We define X_{ijt} a binary variable which equals 1 if O_{ij} is played at time t, 0 otherwise. The first two constraints have been explained previously, The following one links starting times to X_{ijt} variables. The three next ones ensure respectively that each operation is scheduled at some time point, that precedence constraints between operations are met and finally that over all operations, only one can start at a time.

$$\min \sum_{i=1}^{n} \sum_{j=2}^{n(i)} W_{ij}$$

$$\begin{cases} W_{ij} \geq \alpha_i(l_i - S_{ij} + S_{ij-1}) & 1 \leq i \leq n, 2 \leq j \leq n(i) \\ W_{ij} \geq \beta_i(S_{ij} - S_{ij-1} - l_i) & 1 \leq i \leq n, 2 \leq j \leq n(i) \\ S_{ij} = \sum_{t=0}^{H} t X_{ijt} & 1 \leq i \leq n, 1 \leq j \leq n(i) \\ \sum_{t=0}^{H} X_{ijt} = 1 & 1 \leq i \leq n, 1 \leq j \leq n(i) \\ S_{ij} + p_i \leq S_{ij+1} & 1 \leq i \leq n, 1 \leq j \leq n(i) \\ \sum_{i=1}^{n} \sum_{j=2}^{n(i)} \sum_{t'=t-p_i+1}^{t} X_{ijt'} \leq 1 & 1 \leq i \leq n, 1 \leq j \leq n(i), 0 \leq t \leq H \\ 0 \leq S_{ij} \leq H - p_i & 1 \leq i \leq n, 1 \leq j \leq n(i) \\ 0 \leq W_{ij} & 1 \leq i \leq n, 2 \leq j \leq n(i) \end{cases}$$

Both programs give the optimal solution but run with prohibitive computation time (several days/weeks). Still, these programs allow us to compute lower bounds when relaxing the integrity of the binary variables $X_{(i'j')(ij)}$ or X_{ijt}.

5.2 Experiments

Out of our 27 instances, method *LB Time Indexed* gave better results than *Method LB Disjunctive* 23 times and was equal to method *LB Disjunctive* 4 times. The table figure 8 presents for each instance the number of jobs, the total number of operations in the sequence, the load $(\sum_i n(i)p_i)/H$, and then the results obtained with our lower and upper bounds after 1.5 seconds of simulation.

Instances	n	$\sum n(i)$	Load %	LB Disjunctive	LB Time Indexed	Upper Bound
1	9	57	67	24750.00	26370.00	27000.00
2				43350.00	45600.00	46500.00
3				2475.00	2637.00	2700.00
4	14	44	44	0.00	0.00	0.00
5				0.00	0.00	240.00
6				0.00	0.00	78.00
7		62	70	2910.00	3580.00	3840.00
8				5590.00	6400.00	6880.00
9				29.10	35.80	51.30
10		83	99.5	27590.00	32480.00	33500.00
11				47410.00	53680.00	55000.00
12				275.90	324.80	335.00
13	15	91	85	6490.00	7168.78	11670.00
14				5469.40	6169.20	14466.95
15				5469.40	6169.20	81964.10
16				64.90	71.69	4598.30
17				64.90	71.69	55162.55
18		93	90	6490.00	7168.78	12240.00
19				5469.40	6169.20	15100.70
20				5469.40	6169.20	82883.50
21				64.90	71.69	5031.90
22				64.90	71.69	67370.50
23		96	100	6490.00	7229.67	14720.00
24				5469.40	6169.81	19360.65
25				6169.81	6169.81	231675.45
26				5469.40	6169.81	9975.25
27				64.90	72.30	170312.6

Figure 8: Cost Values for LB and UB

6 Conclusion

We have introduced an efficient local search method, based on a compact and efficient neighborhood, to schedule an airborne radar and we have developped lower bounds to test the efficiency of the schedules. Our local search method outperforms all other heuristics. Although it is extremely fast, it still requires too much time to be used in practice. We are currently trying to improve the method by reducing the search space. We also wish to develop simple heuristics that compute competitive schedules.

Acknowledgements

The authors would like to thank E. Chamouard and L. Lupinski for many enlightening discussions on airborne radars.

References

[1] E. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization*, John Wiley & Sons, New York, 1997.

[2] F. Barbaresco. Approche Cognitive de la Gestion Radar *Proceedings of the conference Cogis 2003.*

[3] P. Brucker, J. Hurink, and F.Werner. Improving local search heuristics for some scheduling problems - Part I. *Discrete Applied Mathematics* 65, 97-122, 1996.

[4] P. Brucker, J. Hurink, and F.Werner. Improving local search heuristics for some scheduling problems - Part I. *Discrete Applied Mathematics* 72, 47-69, 1997.

[5] M. Elshafei, H. D. Sherali, J. C. Smith. Radar pulse interleaving for multi-target tracking. Naval Research Logistics Volume 51, Issue 1 , Pages 72–94, 2003.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[7] GNU Linear Programming Kit home page. http://www.gnu.org/software/glpk /glpk.html

[8] J.-P Hardange, Ph. Lacomme and J.-C. Marchais. *Airborne and Spaceborne Radars.* Masson, 1995.

[9] M. Minoux. *Programmation Mathématique, Théorie et algorithmes*, Collection Technique et Scientifique des Télécommunications. Bordas & CNET-ENST, Paris, France, 1983.

[10] R. Möhring, A. Schulz, F. Stork and M. Uetz. Solving Project Scheduling Problems by Minimum Cut Computations. *Management Science*, vol. 49, No. 3:330–350, march 2003.

[11] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*, John Wiley, New York, 1998.

[12] A. J. Orman, C. N. Potts. On the Complexity of Coupled-task Scheduling. *Discrete Applied Mathematics* 72(1-2): 141–154 (1997).

[13] Y. Hendel, Contributions à l'ordonnancement juste à temps, rapport de thèse, université Pierre et Marie Curie, laboratoire Lip6/SysDef, 2005.

[14] G. W. Stimson. *Introduction to Airborne Radar*, 2^{nd} Edition. SciTech Publishing, Inc., 1998.