Cognitive Radio Resource Management Based on Machine Learning

Raviraj S. Adve
Department of Electrical and Computer Engineering, University of Toronto
10 King's College Road, Toronto, ON, M5S 3G4 Canada
e-mail: rsadve@comm.utoronto.ca

Summary

A multifunction radar is designed to perform several tasks which include tracking multiple targets and surveillance of different regions for target detection. Each of these tasks has a desired execution time, a deadline, and an execution period. Often, there are more tasks than can be accommodated or the tasks' desired execution times clash. Therefore, effective radio resource management requires a scheduler to decide on how the available time resources of the radar is allocated to the different tasks at hand. With recent advances in radar technology, multiple channels (different frequencies) can be used to perform multiple tasks at the same time. While this greatly enhances the ability to execute tasks, this also brings up a new problem of scheduling tasks on multiple timelines. Optimal task selection and scheduling is an NP-hard problem on a single timeline, and adding multiple timelines adds another layer of exponential complexity.

Task scheduling for parallel machines (multiple timelines in our case) has been addressed in a number of works in the Operations Research literature. Several heuristics and optimal solutions such as the branch-and-bound method have been proposed. However, the problem of radar resource management with multiple timelines has some specific characteristics that have not been studied yet. Most importantly, unlike in most works in the literature, each task has a deadline, for example a tracker requires an update before a deadline or the track is lost. The task must be dropped after the deadline (with an associated cost and the creation of possibly new tasks).

Our previous work focused on implementing heuristics and comparing the performance obtained with the optimal solution obtained via the branch-and-bound algorithm. Unfortunately, our results showed that the heuristics, while extremely computationally efficient, suffer from a large performance gap with respect to the optimal solution. On the other hand, the branch-and-bound algorithm is computationally feasible only for a small number of tasks.

This paper presents the use of machine learning techniques to achieve performance close to the optimum with low complexity. We use neural networks to augment and speed up the branch-and-bound method. Specifically, at every node of the search tree, a neural network is used to predict the value of the node, and if too far from the optimal solution, the node and its sub branches are eliminated from the tree. The neural network itself is trained using labelled samples which are obtained by executing the branch-and-bound method offline for several thousand sample

problems. Our preliminary results show that the use of neural networks in conjunction with the branch-and-bound method significantly reduces the computational complexity, while the resulting solution is very close to the optimal one.

1. Introduction

A modern radar may be designed to perform multiple functions, such as surveillance, tracking, and fire control. Each function requires the radar to execute a number of transmit-receive tasks. This raises the problem of assigning radar resources, such as the time, frequency and energy budget, to different tasks. Specifically, a radar resource management (RRM) module makes decisions on parameter selection, prioritization, and scheduling of such tasks [1]. RRM becomes especially challenging in overload situations, where some tasks may need to be delayed or even dropped.

Effective resource management for multifunction radars (MFR) has been addressed in the literature (see for example [2]–[4] and references therein). Each radar task comprises transmission, waiting, and reception intervals. RRM, specifically task scheduling, is, in general, an NP-hard problem. As a common theme in the literature, RRM is split into two steps [1]: first, task parameters such as the priority, dwell time, and revisit interval are determined, such as by using rule-based methods for each task individually [5] or by joint optimization, across all tasks, of an overall utility function, while accounting for the resource constraints [2], [6]. Once the parameters are determined, task selection and scheduling is performed. In the selection phase, based on resource constraints and a figure of merit, a subset of the tasks are chosen and the rest dropped. The scheduling phase then assigns time slots to each task. These two steps can also be repeated iteratively to improve performance.

In previous works, the radar functions are performed on a single frequency channel or "timeline". However, with *multichannel*, e.g., multi-frequency, radars becoming increasingly viable, the channels can be used to execute multiple tasks *simultaneously* [7]. A key difference in this paper is that we consider the problem of joint resource management for a multichannel, multifunction, radar, i.e., one that is able to concurrently execute multiple tasks. We ensure that tasks are not scheduled in the interval between transmission and reception. Interleaving methods can also be studied to investigate the potential of more efficiently using the waiting times [8].

In considering RRM for multichannel radars, heuristic methods as well as the optimal branch-and-bound (B&B) technique are developed in [9]. It is shown that the heuristic methods have low performance, and the B&B algorithm can have high computational complexity. In this paper, we propose to use neural networks to boost the B&B method to converge to a solution close to the optimal one while significantly reducing the computational burden. The neural networks are trained using supervised learning. The training datasets are obtained by running the B&B algorithm offline for several thousand sample problems.

2. Problem Formulation

We consider N tasks to be executed on K identical channels within a time window T. Each channel is associated with a timeline. Tasks can be performed in parallel on different channels, but cannot overlap on a given timeline. Once a task has started executing, it cannot be stopped. The duration

(also length or dwell time) of the n-th ($1 \leq n \leq N$) task is denoted by ℓ_n. For each task, there is a *starting time* s_n after which the task is ready to be executed. There is also a *deadline* d_n after which the task cannot be scheduled, and, if not scheduled, must be dropped (with an associated *dropping cost* D_n). For example, consider a tracking task. The starting time of the task depends on the required tracking accuracy and the time when the last measurement was made. The deadline depends on the estimated trajectory of the target and the beamwidth of the radar. The task is dropped after the target is assumed to have moved out of the radar beam. The dropping cost depends on the priority of the task and the further actions required to compensate for the dropping.

A scheduled, but delayed, task suffers a *tardiness cost* which is, here, modeled as linearly proportional to the delay; let e_n be the time when task n begins execution; the tardiness cost is given by $w_n(e_n - s_n)$, where w_n is the weight which scales the delay. Let the binary variable x_n indicate if task n is scheduled (= 1) or dropped (= 0). Then, the cost associated with the n-th task is given by $x_n w_n(e_n - s_n) + (1 - x_n)D_n$. Our joint task selection and scheduling problem is a minimization of the total cost C, given by

$$C = \sum_{n=1}^{N} x_n w_n(e_n - s_n) + (1 - x_n)D_n. \tag{1}$$

Here, the optimization variables are if a task is selected (choosing x_n) and, if $x_n = 1$, the timeline (out of K) on which the task is executed. The final variable is the execution time $e_n (\geq s_n)$. Our optimization problem is, therefore,

$$\begin{aligned} \{x_n^*, e_n^*\} = \; & argmin \sum_{n=1}^{N} x_n w_n(e_n - s_n) + (1 - x_n)D_n \\ s.t.\; & x_n \in \{0,1\}, \quad n = 1, \dots, N \\ & s_n \leq e_n \leq d_n, \quad n = 1, \dots, N \\ & \text{and no tasks overlap in time.} \end{aligned} \tag{2}$$

The scheduling problem without deadlines (and therefore without dropped tasks) is NP-hard [10]. It can be shown that the joint task selection and scheduling is also NP-hard.

3. Proposed Method

Our proposed method is based on the B&B method as introduced in [9]. Here, we augment the B&B algorithm with a neural network which is trained offline from previous executions of the B&B method.

The B&B procedure implicitly enumerates all possible solutions by considering partial solutions in a tree structure. Each partial solution is a sequence of tasks. A schedule is obtained using the "sequence to schedule mapping" [9]. The node at the root is an empty sequence. Nodes in the tree generate children by partitioning the solution space into smaller regions (branching). Each branch is associated with an unscheduled task in the parent node whose deadline has not passed yet. The node at the end of a branch is a new sequence which is obtained by appending the task of the branch at the end of the sequence of the parent node.

A depth-first search strategy is used to traverse the nodes of the tree. *Bounds and dominance rules* are used to prune off nodes that are provably suboptimal (bounding). When a node is pruned off, all of its children nodes are eliminated from the search. Once the entire tree has been explored, the best solution found in the search is returned. The details of the implementation of the B&B algorithm is given in [9]. Importantly, an upper bound *UB* holds the cost of the best complete solution obtained during the search.

The main drawback with the B&B method is that execution time, which could be exponential in the number of tasks. When using dominance rules, the execution time becomes dependent on the task parameters and, in our experience, becomes heavy-tailed, i.e., while many executions are possible in a reasonable amount of time, a few cases require an inordinate execution time.

In order to reduce the complexity of the B&B method, we propose to use a neural network which can predict the least final cost of a complete schedule obtained based on the partial schedule of a given node. Such a prediction is then compared with *UB*, and if it is large enough, the node is eliminated from the search tree. In this way, the B&B method is made faster by removing the nodes that are less likely to end up with the optimal solution.

Each node of the tree represents a partial schedule, where a set of tasks have been scheduled on certain timelines with known execution times and delay costs. Furthermore, a set of tasks are dropped (with certain dropping costs) since their deadlines have already passed on all timelines, and the rest of the tasks are not scheduled or dropped yet. A *state*, s, is defined as a representation of a partial schedule (node). It includes the initial parameters of the tasks as well as their state (scheduled, dropped, or not scheduled) in the corresponding partial schedule.

Given a state s, the optimal value function $v^*(s)$ determines the least overall cost (of a complete schedule) that can be obtained starting from state s. The depth of the search may be reduced by truncating the tree at state s and replacing the subtree below s by an approximate value function $v(s) \approx v^*(s)$. We propose to use a neural network (which we refer to as the value network [11]) to produce the approximate value function. The value network has a state as its input and outputs an approximation of the optimal value of the given state. The weights of the network can be found by regression on the state-outcome pairs $(s, v^*(s))$ obtained from training data. Such training data are, in turn, obtained by the offline execution of the B&B method.

The value network is implemented using a convolutional neural network with 6 layers. The first three layers are convolutional and the last three layers are fully connected. At each convolutional layer, the input is convolved with a filter to produce the output. The coefficients of the filter are obtained using supervised training.

The output of each layer goes through a non-linear function (the rectifier function) before being passed to the next layer. The final output of the network is a scalar number representing the estimated least overall cost of the partial schedule input. The input to the network (a partial schedule) is a matrix with each column representing a task and each row representing a feature of the corresponding task. One input feature is dedicated for each timeline (channel), and we use one-hot encoding to represent the channel on which the task is scheduled. For each task, the input feature corresponding to the channel on which the task is scheduled is set to 1, and the rest of timeline input features are set to 0. As an example, assuming that there are 4 timelines (channels)

available for task scheduling, each task will have 4 features to represent the assigned timeline (only one input corresponding to the assigned timeline is set to 1, and the rest are 0).

In this paper, we have considered 14 features for each task as given in Table 1. It is assumed that there are 4 channels available for task scheduling.

Table 1. Features of the tasks as used for the input of the value network.

Input feature	Description
1, 2, 3	status (1 0 0: scheduled, 0 1 0: dropped, 0 0 1: unscheduled)
4	start time
5	end time
6	task length
7	execution time
8	tardiness coefficient
9	dropping cost coefficient
10	tardiness cost or drop cost (if scheduled or dropped)
11, 12, 13, 14	assigned timeline (one-hot encoded)

For the convolutional layers, filters with a width of 7 (looking at the features of 7 consecutive tasks at each stride) are used. At each layer 64 filters are used (the output of each convolutional layer has 64 features).

We have considered 512 hidden units for the first fully connected layer. The second fully connected layer has 128 hidden units, and the last layer has one scalar output. The network is trained using 90000 samples obtained from the branch-and-bound method. The weights are obtained by minimizing the L2-loss using the Adaptive Moment Estimation (Adam) optimization method [12]. We use 100000 steps of the Random Reshuffling (RR) method [13] with mini-batches of size 100.

At each node of the B&B algorithm, we use the value network to obtain the predicted least final cost that can be achieved from that node. If the estimated value is larger than the cost of the best solution found so far, UB, multiplied by a scaling coefficient β (i.e. the estimated value $\geq \beta \times UB$), the node is eliminated from the search. The scaling coefficient $\beta \geq 1$ is introduced to make the algorithm robust to estimation errors. Better performance may be achieved with larger values of β, but that would mean that less nodes are eliminated from the search tree.

4. Simulation Results

We now present the results of simulations illustrating the performance of the proposed method. We consider a multichannel radar with $K = 4$ identical channels. There are $N = 35$ number of tasks distributed over a timeline window of 100 sec. The objective is to schedule the tasks such that the overall cost of dropping and delaying the tasks, the objective in (2), is minimized. In our simulations, the starting time of the tasks is uniformly distributed on the 100 sec time window, i.e. $s_n \sim U(0, 100)$. ($x \sim U(a, b)$ represents a uniform random variable distributed between a and b).

For each task, the interval between the starting time and the deadline, i.e. $(d_n - s)_n$, is sampled from $U(2, 12)$. The task length ℓ_n is distributed according to $U(2, 11)$. Furthermore, the dropping costs (D_n) and the tardiness weights (w_n) have distributions $U(100, 500)$ and $U(1, 5)$, respectively. In the simulations, we use the Monte Carlo method with 100 trials to obtain the average performance of each method. We emphasize that these models for the parameters is for simulations only; in practice, the parameters would be determined by mission requirements.

The average cost of the heuristic methods, the B&B algorithm, and the proposed method with different scaling coefficients is given in Table 2. The B&B method has the best performance and highest complexity. The average number of visited nodes in the search tree for the B&B algorithm is 13134. In case that the value network with $\beta = 1$ is used, the average number of visited nodes drops to only 448 nodes which is about 30 times less than the number of visited nodes of the B&B method without the value network. While the value network significantly reduces the complexity of the B&B method, the degradation in the performance is not much, and the proposed method has better performance than the heuristic algorithms. The performance of the B&B method with the value network can be improved by using a larger value for the scaling coefficient β. However, it would increase the number of visited nodes in search tree. In the case of $\beta = 1.5$ and $\beta = 2$, the average number of visited nodes are 1466 and 2460, respectively.

Table 2. Average cost of different methods for scheduling *N*=35 tasks over *K*=4 channels. EST and ED refer to earliest starting time first and earliest deadline first methods, respectively [9]. SW refers to the task swapping technique [9]. The last three columns are the results of the B&B method when enhanced by the value network with different scaling coefficients.

	EST	EST+SW	ED	ED+SW	B&B	$\beta = 1$	$\beta = 1.5$	$\beta = 2$
Average Cost	93.2	68.3	115.8	101.5	38.6	45.7	44.5	42.9

Finally, in Table 3., we compare the complexity and performance of the proposed method with the B&B algorithm for a few sample instances. The performance of the best heuristic method is also given for comparison.

Table 3. Cost and number of visited nodes (cost, #node) for the B&B and enhance B&B with the value network with different scaling coefficients as well as the cost of the best heuristic method.

	B&B	$\beta = 1$	$\beta = 1.5$	$\beta = 2$	EST+SW
Sample 1	(54.9, 1670)	(63.1, 36)	(63.1, 52)	(63.1, 211)	63.1
Sample 2	(73.6, 554298)	(96.3, 110)	(96.2, 1512)	(78.9, 54027)	100.0
Sample 3	(221.9, 54514)	(221.9, 23610)	(221.9, 46626)	(221.9, 52705)	328.4
Sample 4	(47.5, 6998)	(49.4, 173)	(49.4, 196)	(47.5, 206)	49.7
Sample 5	(30.1, 2017)	(39.1, 105)	(39.1, 131)	(39.1, 153)	39.1

4. Conclusion

In this paper, we show how the complexity of the branch-and-bound method can be reduced using machine learning methods. Specifically, a value network is used at each node of the search tree to predict the least final cost that can be achieved from the given partial schedule of the node. While this significantly reduces the number of visited nodes of the search tree, the performance does not suffer from much degradation. The value network is trained using supervised learning with data samples obtained from the offline execution of the optimal B&B method.

References

[1] P. W. Moo and Z. Ding, *Adaptive Radar Resource Management*. Academic Press, 2015.
[2] A. Charlish, K. Woodbridge, and H. Griffiths, "Phased array radar resource management using continuous double auction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 51, no. 3, pp. 2212–2224, Jul. 2015.
[3] S. L. C. Miranda, C. J. Baker, K. Woodbridge, and H. D. Griffiths, "Comparison of scheduling algorithms for multifunction radar," *IET Radar Sonar Navig.*, vol. 1, no. 6, pp. 414–424, Dec. 2007.
[4] A. M. Ponsford, L. Sevgi, and H. C. Chan, "An integrated maritime surveillance system based on high-frequency surface-wave radars, Part 2: Operational status and system performance," *IEEE Antennas Propag. Mag.*, vol. 43, no. 5, pp. 52–63, Oct. 2001.
[5] S. P. Noyes, "Calculation of next time for track update in the MESAR phased array radar," in *IEE Colloquium on target tracking and data fusion (digest no. 1998/282)*, Jun. 1998, pp. 2/1–2/7.
[6] A. Sinha, Z. J. Ding, T. Kirubarajan, and M. Farooq, "Track quality based multitarget tracking algorithm," in *Proc. SPIE 6236, Signal and Data Processing of Small Targets 2006, 623609*, May 2006.
[7] J. Yan, H. Liu, B. Jiu, B. Chen, Z. Liu, and Z. Bao, "Simultaneous multibeam resource allocation scheme for multiple target tracking," *IEEE Trans. on Sig. Proc.*, vol. 63, no. 12, pp. 3110–3122, June 2015.
[8] H. S. Mir and A. Guitouni, "Variable dwell time task scheduling for multifunction radar," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 463–472, Apr. 2014.
[9] M. Shaghaghi and R. S. Adve, "Task selection and scheduling in multifunction multichannel radars," to appear in *Proc. IEEE Inter. Radar Conf.*, Seattle, May 2017.
[10] J. K. Lenstra, A. R. Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics*, vol. 1, pp. 343–362, 1977.
[11] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
[12] D. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[13] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo, "Why random reshuffling beats stochastic gradient descent," arXiv preprint arXiv:1510.08560, 2015.