Data Parallelism in Traffic Control Tables with Arrival Information

Juan F.R. Herrera[1], Eligius M.T. Hendrix[2], Leocadio G. Casado[1], and René Haijema[3]

[1] University of Almeria (ceiA3), Spain
{juanfrh,leo}@ual.es
[2] Universidad de Málaga, Spain
Eligius.Hendrix@wur.nl
[3] Wageningen University, The Netherlands
Rene.Haijema@wur.nl

Abstract. Traffic lights can be controlled dynamically through rules reacting on the number of waiting vehicles at each light. A rule can be captured by a so-called Traffic Control Table (TCT). The Value Iteration method from Stochastic Dynamic Programming has been used for simple networks to derive a TCT. This work studies the generation of a TCT-based rule that takes the arrival information of new vehicles into account. The question is how to generate such a table for simple intersections (or a network of these). The generation is particularly difficult due to the computational work involved in the Value Iteration process.

The problem is formulated as a Markov Decision Process and the parallelization of the Value Iteration method for this problem is discussed. We are specifically interested in exploiting the structure of the problem for simple infrastructures, with only a few traffic lanes, using a parallel algorithm.

Keywords: Markov Decision Process, Stochastic Dynamic Programming, Value Iteration, Traffic Control.

1 Introduction

Dynamic decision making represents a wide area in real life, where the main goal is taking a decision from a certain state, i.e., decide the action x being in state s and time t. Here, we consider the problem of minimizing the waiting time at traffic lights.

Traffic lights were introduced in the early twentieth century to make traffic safer in places where traffic from different directions intersects at what is called a junction or intersection. To give priority to several traffic flows, the vehicles approaching from other flows must wait before getting right of way. The overall waiting time can be reduced if there exists a conveniently controlled sequence in the traffic lights network. This problem has been considered by theorists as well as engineers [2, 6–8].

© Springer International Publishing Switzerland 2014

The optimal control traffic in a traffic junction is based in what we call a Traffic Control Table (TCT). This table determines which flow combination has priority given the current traffic state (number of waiting vehicles in each queue) in the traffic junction. To find a TCT which minimizes the vehicle waiting time, one can model the problem as a Markov Decision Process (MDP) and apply a Value Iteration algorithm, based on backward induction [3–5]. The idea of backward induction was introduced by Bellman in 1953 to solve (stochastic) dynamic programming [1]. Backward induction can be applied in dynamic optimization problems where time is discrete, for solving stationary systems without a time horizon.

Dynamic programming usually deals with a considerable number of states. Executing the Value Iteration algorithm requires to store all this information in main memory. An efficient access to these values can facilitate the cache location and therefore save computational time. In this work, the study of parallelizing the Value Iteration algorithm is tackled taking the data management into account.

The paper is structured as follows. In Section 2, a MDP model is described for the decisions captured in a TCT. Section 3 provides the algorithm of Value Iteration for the described problem. Section 4 applies the method to a simple infrastructure. The parallel algorithm as well as the experimental results will be shown in Sections 5 and 6 respectively. Conclusions and future work are presented in Section 7.

2 Model Description

The main goal is to calculate a TCT that determines how to manage the traffic lights considering the traffic density, so that overall waiting time is minimized in the long run. The table should capture which flows should get green light given the current state of the traffic. The problem is formulated as a MDP.

2.1 Goal

The waiting time of a vehicle is defined as the time from the moment a vehicle joins the queue until it crosses the stopping line. The practical objective is to find a control that minimizes the Expected Waiting (EW) time in the long run. Little's law formalizes the relationship between the arrival rate λ, EW time, and the number of waiting vehicles in a queue (EQ) as follows:

$$EQ = \lambda \times EW.$$

If the number of vehicles waiting in the queues is minimized, the waiting time is also minimized. The MDP algorithm does not need to keep track of the time a particular vehicle is waiting, but it does so for the number of queued vehicles.

2.2 Model Assumptions

The optimization problem of determining the traffic lights control is modelled as a stochastic and discrete-time control process. Given the state s of the traffic and the lights, an action x must be chosen from the available ones. The optimal choice $x(s)$ is then derived via the process of Value Iteration.

The model is based on the following ingredients:

- Time is divided into slots.
- A slot is the time required for a vehicle to leave a queue and cross an intersection. It also determines the safety distance between two vehicles.
- Traffic lights can be changed only at the end of a slot.
- The change from green to red light requires two slots of yellow: *Y1* and *Y2*.
- The decision of changing the traffic light state is implemented instantly.
- While the light is green or yellow, at most one vehicle can cross the intersection in a slot.
- Vehicles move at a constant speed. The queued vehicles do not move, except for the first queued vehicle when the light allows it.
- To facilitate the analysis, the vehicle length is neglected. Therefore, a traffic junction is never stuck due to traffic.

2.3 Summary of Notations

The notation used throughout this paper is defined below. A crossing or intersection consists of F traffic flows or lanes, where each flow f consists of a lane and an associated queue. Vector $q = (q_1, \ldots, q_F)$ stores the number of vehicles waiting in each queue. Although in reality queues may contain Q or more vehicles, queues are truncated at Q vehicles to facility numerical computation of an optimal policy. We will also consider the possibility of having arrival information in a vector $a^{(f)}$ as will be specified in the next section. The traffic lane set $\mathcal{F} = \{1, \ldots, F\}$ can be partitioned in C disjoint subsets called combinations. Conflicting traffic flows do not get green simultaneously. The various combinations are given beforehand and determine the instance of the problem. All the lanes within a combination simultaneously receive green, red or yellow light. If a combination receives green or yellow light, the remaining ones receive red light.

2.4 Formulation as a Markov Decision Process

To model the problem, the following parameters are used:

- $\mathcal{S}$: a finite set of states. A state is defined by the light and the number of vehicles waiting in the queues that define the intersection. The number of states is countable because the number of intersections as well as the queue length are limited. The queues associated to each intersection are artificially truncated at some maximal queue length. Arrival information of new vehicles is also considered in state s.
- $c(s)$: cost of state s, defined by the number of vehicles waiting in the queues that describe the state s.

- λ_f: the probability that a new vehicle joins traffic flow f in a slot and therefore $(1-\lambda_f)$ is the probability that no vehicle joins the traffic flow.
- $\mathcal{E}$: a finite set of possible events. Each event represents the arrival or not of a new vehicle at a lane f. Notice that the event does not depend on the state.
- $\mathcal{X}(s)$: a finite set of possible actions to control the lights. The set of possible actions depends on the state s.

State s: For the dynamic traffic control, the state of the MDP is the current light situation and state of traffic flows. The possible light states are denoted by index $l \in \mathcal{L} = \{1, \ldots, L\}$, where $L = 1 + 3^C$ as either all lights are red or only one of each of the C combinations has green, *Y1* or *Y2*. The state of traffic flows for this model is:

- q : The number of vehicles waiting in each lane.
- $a^{(f)} = (a_1, \ldots, a_M)^{(f)}$: M arrival information slots are available for lane f, where $a_i^{(f)} \in \{0, 1\}$ determines whether a car arrives at queue f in i time slots from now.

Both the maximum number of vehicles in a single queue Q and the number of arrival information slots M may be flow specific.

Therefore, a state s is defined by $s = (l, q, a)$. The number of possible states is given by

$$|\mathcal{S}| = |\mathcal{L}| \cdot (Q+1)^F \cdot 2^{M \cdot F}. \tag{1}$$

Control Action x: Given a state s, the action $x \in \mathcal{X}(s) \subseteq \mathcal{L}$ immediately adjusts the lights.

State Transition: During a slot, one vehicle arrives at traffic flow f with probability λ_f. The stochastic event e is defined as a vector of F elements $e = (e_1, \ldots, e_F)$, where $e_f \in \{0, 1\}$ denotes the number of vehicles that enter the infrastructure at lane f, within the coming time slot. Let function $\Delta_f(x)$ denote whether lane f has right of way ($\Delta_f(x) = 1$) or not ($\Delta_f(x) = 0$) when the lights are changed due to action x. As a result of decision x, state (l, q, a) changes into state

$$T(x, s_j, e_i) = (x, (q_1 + a_1^{(1)} - \Delta_1)^+, (a_2^{(1)}, \ldots, a_M^{(1)}, e_1), \ldots, (a_2^{(F)}, \ldots, a_M^{(F)}, e_F)).$$

Objective Function: The objective function is to minimize the number of vehicles waiting at the queues. The contribution to the objective function over a single time slot is $c(s) = \sum_f q_f$. The associated cost in a general MDP usually depends on the state s as well as the decision x. In the model we are interested in, it depends on the state only, so we have the cost function $c(s)$.

Bellman's Principle of Optimality: The strategy $x(s)$ is optimal [1] if there exists a function $v(s)$ and a scalar d such that $\forall s \in \mathcal{S}$

$$v(s) - d = c(s) + \min_{x \in \mathcal{X}(s)} [E\{v(T(x, s, e))\}], \tag{2}$$

where E symbolizes the expected value with respect to the stochastic event e. The transition function $T(x, s, e)$ describes the next state to be reached after taking the decision x in the state s on event e.

Essential in the described model is that the number of events is finite, such that they can be numbered as e_i with probability of occurrence p_i. In addition, we also consider that the countable state space is finite, such that the states are indexed $j = 1, \ldots, |\mathcal{S}|$. This implies that the value function v can be captured by a vector V with elements $V_j = v(s_j)$. The Bellman equation (2) implies to find a valuation vector V and a constant d such that

$$V_j - d = c(s_j) + \min_{x \in \mathcal{X}(s_j)} \sum_i p_i V_k\,, \tag{3}$$

where k is the state index value related to state $T(x, s_j, e_i)$. Notice that the index values of the reachable states from state s_j are a subset

$$\mathcal{K}_j = \{k : s_k = T(x, s_j, e_i)\ \forall i, \forall x \in \mathcal{X}(s_j)\}$$

of all states $\mathcal{S}$. If one has a valuation V, also the optimum control value can be derived from

$$X_j = x(s_j) = \arg \min_{x \in \mathcal{X}(s_j)} \sum_i p_i V_k, \tag{4}$$

where again $V_k = T(x, s_j, e_i)$.

3 Value Iteration through Backward Induction

Bellman introduced the term "backward induction" and also indicated the way to solve (3) using fixed point theory, where (3) is repeated iteratively. The latter process is called Value Iteration [9]. One way to deal with that (see Algorithm 1) is to copy a current valuation vector V into a vector W and determine a new valuation V according to

$$V_j = c(s_j) + \min_{x \in \mathcal{X}(s_j)} \sum_i p_i W_k,\ \ j = 1, \ldots, |\mathcal{S}|, \tag{5}$$

where k is the state index related to state $T(x, s_j, e_i)$.

The Value Iteration should lead to convergence towards $d = V_j - W_j$, for all $j = 1, \ldots, |\mathcal{S}|$. Convergence to the scalar d is measured by the so-called

$$\text{span}(V, W) = \max_j (V_j - W_j) - \min_j (V_j - W_j).$$

The iterative procedure of Algorithm 1 stops whenever $\text{span}(V, W)$ is smaller than a pre-specified value ϵ, which indicate the accuracy in estimating d.

Algorithm 1. Value iteration

```
1: Set vector $V$ to zero
2: repeat
3:     Copy vector $V$ into vector $W$
4:     for $j = 1, \dots, |\mathcal{S}|$ do
5:         $V_j = c(s_j) + \min_{x \in \mathcal{X}(s_j)} \sum_i p_i W_k$
6:     end for
7: until $\max_j(V_j - W_j) - \min_j(V_j - W_j) < \epsilon$
```

How is this property translated into the TCT model? In the first place, notice that the strategy $x(s)$ represents the idea of a TCT. In the model, the set of states $\mathcal{S}$ is countable as it is based on the light colours and the number of waiting vehicles. In addition, a maximum number of vehicles in a queue is defined, such that the total number of states becomes finite. The value of states beyond the limit Q are obtained by extrapolation. The function $v(s)$ is represented by vector V and $x(s)$ is the final TCT given by (4). The computational challenge is that $|\mathcal{S}|$ is usually high. An important observation for the defined model is that the computation of V_j requires a small index subset $\mathcal{K}_j$ for values W_k of the index set $\mathcal{S}_j$.

The research question is how to handle the retrieval of W_k values from memory in the computational process, such that the loop over j in Algorithm 1 can be done in an efficient way.

4 Studied Case of the TCT Model

In [3], a code is introduced to denote a specific infrastructure. For instance, I1F2C2 stands for a single intersection with $F = 2$ traffic flows (or lanes) and $C = 2$ combinations. Figure 1 shows the simple infrastructure that will be elaborated:

I1F2C2 is a single T-shape intersection (T-junction) with two traffic flows, $\mathcal{F} = \{1, 2\}$. Vehicles in lane 1 drive from west to east, while vehicles in lane 2 drive from north to east, i.e., they turn left. Each direction has a single lane with its corresponding queue at the stopping line. Additionally, lane 1 has arrival information slots, represented by the vector a.

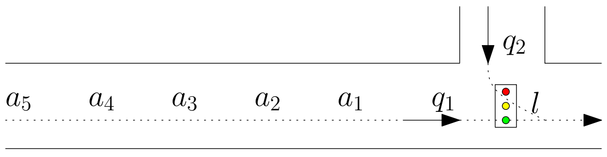

Fig. 1. I1F2C2 infrastructure

The preliminary results focus on this infrastructure. The specific features of the I1F2C2 case are as follows.

State s: A state s is defined by $s = (l, q, a)$, see Section 2. The traffic light states are

$$\mathcal{L} = \{0, 1, 2, 3, 4, 5, 6\} = \{\mathit{Red}, \mathit{GF1}, \mathit{GF2}, \mathit{Y1F1}, \mathit{Y2F1}, \mathit{Y1F2}, \mathit{Y2F2}\},$$

where:

0. *Red:* Red light for all the flows.
1. *GF1:* Green light for flow 1.
2. *GF2:* Green light for flow 2.
3. *Y1F1:* Yellow light for flow 1 (slot 1).
4. *Y2F1:* Yellow light for flow 1 (slot 2).
5. *Y1F2:* Yellow light for flow 2 (slot 1).
6. *Y2F2:* Yellow light for flow 2 (slot 2).

The state of traffic flows for this case is:

– $q = (q_1, q_2)$. The number of vehicles waiting in each lane.
– $a = (a_1, \ldots, a_M)$. As only flow 1 has arrival information, the flow index is omitted. The element $a_t \in \{0, 1\}$ determines the presence or absence of a vehicle in slot t. For example, if $a_t = 1$, a vehicle arrives to the queue in t time slots.

The number of possible states for the I1F2C2 case is given by

$$|\mathcal{S}| = |\mathcal{L}| \cdot (Q + 1)^2 \cdot 2^M.$$

Control Action x: The possible subsets $\mathcal{X}(s) \subset \mathcal{L}$ are enumerated as follows.

$$\mathcal{X} = \begin{cases} \{1,2\} = \{\mathit{GF1}, \mathit{GF2}\} & \text{if } l = 0 = \mathit{Red} \\ \{1,3\} = \{\mathit{GF1}, \mathit{Y1F1}\} & \text{if } l = 1 = \mathit{GF1} \\ \{4\} \quad = \{\mathit{Y2F1}\} & \text{if } l = 3 = \mathit{Y1F1} \\ \{0\} \quad = \{\mathit{Red}\} & \text{if } l = 4 = \mathit{Y2F1} \\ \{2,5\} = \{\mathit{GF2}, \mathit{Y1F2}\} & \text{if } l = 2 = \mathit{GF2} \\ \{6\} \quad = \{\mathit{Y2F2}\} & \text{if } l = 5 = \mathit{Y1F2} \\ \{0\} \quad = \{\mathit{Red}\} & \text{if } l = 6 = \mathit{Y2F2} \end{cases}$$

State Transition: As I1F2C2 has only two flows, the event is defined as a vector e of two elements $e = (e_1, e_2)$. For this simple case of two lanes, four possible events can happen: $e \in \{(0,0), (0,1), (1,0), (1,1)\}$. The probability p_i related to the events is:

$$\begin{cases} i = 1,\, e = (0,0) \text{ with } p_1 = (1 - \lambda_1)(1 - \lambda_2) \\ i = 2,\, e = (0,1) \text{ with } p_2 = (1 - \lambda_1)\lambda_2 \\ i = 3,\, e = (1,0) \text{ with } p_3 = \lambda_1(1 - \lambda_2) \\ i = 4,\, e = (1,1) \text{ with } p_4 = \lambda_1\lambda_2 \end{cases}$$

For instance, let $M = 5$. During a slot, the arrival information denoted by $a = (a_1, a_2, a_3, a_4, a_5)$ shifts towards $a \leftarrow (a_2, a_3, a_4, a_5, e_1)$, i.e., the arriving vehicles at the lane are approaching and a_1 represents the vehicle that is added to the queue.

Depending on the traffic light state, the queue length will increase or not. Suppose that flow 1 has red light ($l \in \{0, 2, 5, 6\}$, so $\Delta_f(x) = 0$). A vehicle which is one slot upstream away from queue f, increases the number of vehicles in queue 1. When x sets the colour to green or yellow for lane 1 ($l \in \{1, 3, 4\}$, so $\Delta_f(x) = 1$) and the queue 1 is not empty ($q_1 > 0$), the vehicle is added to the queue. If the queue is empty ($q_1 = 0$), the vehicle crosses the stop line of flow 1 without delay. The transition of queue 1 is given by $q_1 \leftarrow q_1 + a_1 - \Delta_1(x)$.

For lane 2, there exists no information about the arrival of new vehicles to the queue, therefore the next state of queue 2 depends on e_2 and action x. The transition of queue 2 is given by $q_2 \leftarrow (q_2 + e_2 - \Delta_2(x))^+$ vehicles.

Objective Function: Having two queues, the objective function is

$$c(s) = q_1 + q_2 \,.$$

5 Parallel Approach

The sequential Algorithm 1 is an iterative process where the vector V of size $|\mathcal{S}|$ is updated based on previous values of V_j stored in vector W until a termination condition holds. The natural way to tackle the parallelization of Algorithm 1 is to distribute the evaluation of V between processors. The set of state values V can be partitioned in different ways depending on $|\mathcal{S}|$ and on the characteristics of the final architecture. When message passing is used, the number of messages should be as small as possible. For shared-memory architectures, the access to values in W should be done in such a way that the probability they are available in cache level 1 is increased.

Both methods need a synchronization point at the end of the update of V to check the termination condition and to have the data W available for the next iteration. This is an additional challenge for the parallelization of the process.

Taking these considerations into account, an option is to store the values of V and W as a matrix, where the columns are the values of lights and the rows are the different combinations of queue states and arrival information. In a shared-memory architecture, the matrix can be computed row wise, where each thread is in charge of a chunk of the matrix. In a distributed-memory architecture, a reasonable solution is a partition of the matrix into columns. In this way, the number of messages is not high, although the message size may easily be big.

5.1 Computational Complexity

The workload is related to the number of flows (F), the maximum size of a queue (Q) and the number of states (M) that describes the arrival information. The

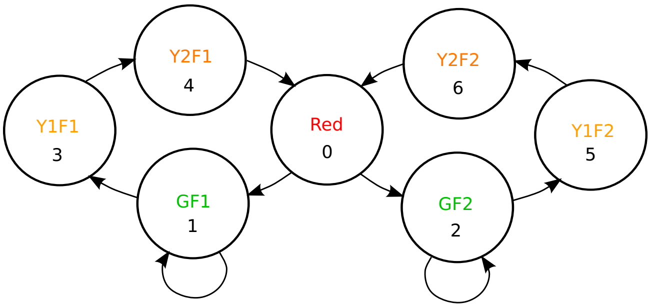

Fig. 2. Message passing in lights distribution for MPI approach

limit imposed on the queue length makes the state space finite. Higher values of queue lengths introduce state situations with an extremely low probability of occurrence. This hinders the convergence of Value Iteration. Therefore, for each iteration the complexity of the algorithm is given by $|\mathcal{S}|$ as specified by (1).

5.2 MPI Approach

The MPI approach is mainly used for distributed-memory systems. Figure 2 shows the message passing for the parallelization using MPI, where each MPI process is in charge of the evaluation of the values of V associated to one value of the light state $l \in \mathcal{L} = \{0, 1, 2, 3, 4, 5, 6\}$. A greater parallel degree can be achieved by parallelizing each MPI process.

The main drawback of the described MPI approach is the imbalance between processes. For instance, a process in charge of $l \in \{Red, GF1, GF2\}$ requires more data from vector W to update V_j values than processes in charge of yellow lights. In addition, the maximum number of processes in I1F2C2 is seven according to the developed strategy.

5.3 Threaded Approach

This approach is used in shared-memory architectures. Having V as a matrix, where each thread is in charge of the evaluation of a set of rows facilitates the use of an arbitrary number of threads. Matrix V is partitioned into chunks of rows, each to be dealt with by one thread. A drawback is that usually the effective number of threads is bounded by the number of cores in the shared-memory architecture.

6 Preliminary Results

Experiments have been carried out in a node of *BullX-UAL* cluster, with two Intel® Xeon® E5-2650 processors and 64 GB of main memory. Algorithms have

Table 1. Numerical results using threads

	Q50M10		*Q100M5*	
N. of threads	Time (s)	Speedup	Time (s)	Speedup
1	240	–	511	–
2	119	2	260	2
4	56	4	133	3.8
8	31	7.7	68	7.5
16	17	14	36	14.2

been coded in C and compiled using gcc with MVAPICH and POSIX threads API.

The arrival ratio is $\lambda_f = 0.2$ for each incoming flow $f = 1, 2$. The accuracy is set to $\epsilon = 0.01$. TCTs are generated for I1F2C2 with two different settings:

Q50M10: Queue length $Q_1 = Q_2 = 50$. Arrival information slots $M = 10$.
Q100M5: Queue length $Q_1 = Q_2 = 100$. Arrival information slots $M = 5$.

6.1 MPI Approach

Results for the MPI approach using seven processes shows a poor performance with a speedup of less than 3 due to the imbalance of the workload among processes and the cost of message passing.

6.2 Threaded Approach

Table 1 shows the experimental results for the threaded version varying the number of threads. The achieved speedup is near linear. As future work we plan to perform a reordering of the rows of value functions V and W assigned to each thread to improve the cache locality of the needed data.

7 Conclusion and Ongoing Work

Solving dynamic optimization problems by the Value Iteration algorithm is a challenge from the point of view of running the process in parallel. Each iteration of the algorithm requires a synchronization. This paper studied the algorithm for generating the best traffic control table. The difficulty of these problems increases with the number of traffic flows and intersections, and the inclusion of arrival information. A near linear speedup has been obtained with a threaded version for a basic instance of the problem. Future investigation will focus on parallel computation of traffic control tables for more complicated infrastructures that imply a larger number of traffic states. To do so, a parallelization in two levels (MPI for inter-node and threads for intra-node) is convenient to achieve a good performance in a cluster of shared-memory nodes.

Acknowledgments. This work has been funded by grants from the Spanish Ministry (TIN2008-01117 and TIN2012-37483) and Junta de Andalucía (P11-TIC-7176 and P12-TIC-301), in part financed by the European Regional Development Fund (ERDF). J.F.R. Herrera is a fellow of the Spanish FPU program.

References

1. Bellman, R.: A Markovian Decision Process. Journal of Mathematics and Mechanics 6(5) (1957)
2. van den Broek, M.S., van Leeuwaarden, J.S.H., Boxma, I.J.B.F.A., Bounds, O.J.: Approximations for the fixed-cycle traffic-light queue. Transportation Science 40(4), 484–496 (2006)
3. Haijema, R.: Solving large structured Markov Decision Problems for perishable inventory management and traffic control. Ph.D. thesis, Universiteit van Amsterdam (2008)
4. Haijema, R., Hendrix, E.M.T.: Traffic responsive control of intersections with predicted arrival times: A Markovian approach. Computer-Aided Civil and Infrastructure Engineering 29(2), 123–139 (2014)
5. Haijema, R., van der Wal, J.: An MDP Decomposition Approach for Traffic Control at Isolated Signalized Intersections. Probability in the Engineering and Informational Sciences 22, 587–602 (2008)
6. van Leeuwaarden, J.S.H.: Delay analysis for the fixed-cycle traffic-light queue. Transportation Science 40(2), 189–199 (2006)
7. Newell, G.F.: Approximation methods for queues with application to the fixed-cycle traffic light. SIAM Review 7(2), 223–240 (1965)
8. Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., Wang, Y.: Review of road traffic control strategies. Proceedings of the IEEE 91(12), 2043–2067 (2003)
9. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edn. John Wiley & Sons, Inc., New York (1994)