The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

Real-Time Planning for Traffic Signal Control

Alison Paredes, Wheeler Ruml
Department of Computer Science
University of New Hampshire
Durham, NH 03824 USA
{alison, ruml} @cs.unh.edu

Abstract

Unusual events, such as sporting events, road works, and natural disasters, can overwhelm city infrastructure such as traffic networks, which have limited capacity. Recent work on automating traffic flow to reduce congestion and its effects, such as pollution, has formulated the problem of responding to sensed congestion using hybrid domain-independent planning. This approach employs complex domain-independent solvers and requires them to formulate a complete multi-step plan to eliminate the congestion before the first response to the congestion can be initiated. In this short paper, we present a much simpler, domain-dependent solver that uses a receding-horizon approach based on real-time heuristic search to select appropriate actions incrementally. Empirical results comparing this approach to previous work are encouraging, demonstrating reasonable behavior and much better scalability. These results suggest that smart cities can leverage real-time planning to control traffic signals on a city-wide scale.

Introduction

The internet of things has the potential to change how cities respond to unexpected events like natural disasters which can easily overwhelm existing infrastructure such as traffic networks. Traffic signals have been automated in the past, however, typical signals are limited to react only in predefined patterns according to local sensors such as inductive loops. This is often adequate to adapt behavior to typical changes in traffic flow. Commuters, for example, can expect traffic signals to work in their favor during rush hour. With the rise of networked sensing and signaling infrastructure, we should now expect intelligent decision-making to be able to take the entire state of the traffic network into account. There has been recent work in automating traffic signal networks using approaches from automated planning, which promise more flexible solutions to atypical events. In the case of unexpected events, these approaches may be better at relieving congestion and avoiding its consequences than predefined solutions. Networked signals have the advantage of being able to observe the current state of the entire network and respond accordingly. Smart cities can combine these kinds of traffic signaling systems with planning to control the entire network in an emergency to promote the most efficient flow in real-time as the state of the network evolves.

There has been significant previous work in this direction already. We take as our starting point the work of McCluskey and Vallati (2017). In a simulated demonstration, they collect data in real time from the road network to assess its current state, compile it into a hybrid planning problem (expressed in the language PDDL+), and then use an off-the-shelf domain-independent hybrid planning system (UPMurphi) to create a plan to relieve any congestion. The time spent planning is on the order of minutes, because their solver is very general and the approach is inherently off-line, that is, it finds a complete plan to the entire problem before executing a single signal control action. While McCluskey and Vallati (2017) demonstrate interesting results on a network with seven signals, their approach is not currently scalable to a city-sized problem or real-time response latencies. Recall that congestion can be caused by a myriad of unanticipated events. Ideally, as soon as congestion is observed, the network should try to alleviate it, using all the information and resources at its disposal, and adapting its actions as the situation evolves. In this short paper, we present a preliminary exploration of the potential of real-time planning as a scalable approach to solving large traffic signal control problems like this.

Other approaches with the potential to scale to city-sized problems have been proposed. SURTRAC (Smith et al. 2013) uses a distributed network of interacting signal agents, and it has been shown in simulation and in the field to reduce delay during normal conditions which include normal changes like rush hour, but it has not been tested on exceptional conditions like we are interested in. Mixed integer linear programming has the potential to be used to optimize travel time over a large network by modeling the problem at a lower resolution but it has not been shown to scale to a city-scale problem (Guilliard et al. 2016).

Real-time planning is promising for two reasons. Firstly, it has been shown to scale not in the size of the problem but in the size of the solution (Korf 1990). In other words, as the number of intersections we want to control grows, the time reserved for planning does not necessarily need to change. SURTRAC's distributed approach seeks to avoid this issue but sacrifices the ability to consider the long-range effect of one signal on another. A real-time approach should allow us to scale to a city-sized problem controlling hun-

dreds of signals. Secondly, real-time planning has the potential to begin moving the network toward normal again in the order of seconds. It is guaranteed to be able to respond with at least a partial plan in a constant amount of time (Koenig and Sun 2009; Kiesel, Burns, and Ruml 2015; Sharon, Felner, and Sturtevant 1992; Sturtevant and Bulitko 2011).

In this paper, we propose that real-time planning is a competitive approach to control a large network of traffic signals in response during an exceptional event. We begin by formalizing the problem of alleviating a traffic network that has become overwhelmed; we show how we formulate this problem as a planning problem and how real-time heuristic search can be applied to it. While our efforts are still preliminary, our first empirical results are encouraging. We show that, given the realistic problem addressed by McCluskey and Vallati (2017), a basic real-time planning approach exhibits good behavior, outperforming a fixed signal policy, and has the potential to scale, returning control decisions within seconds. These promising results suggest that a real-time planning approach is worthy of consideration for intelligent decision-making in smart city infrastructure.

Problem Formulation

In this section we define the key elements of the problem we are interested in and map them to a planning problem.

A traffic network is made up of intersections and the roads (streets, highways, etc.) that link them. The amount of traffic on a link can be measured in terms of passenger units and is represented by a queue. Traffic signals control the flow of passenger units from one queue to another, which is given as a flow rate in terms of the number of passenger units per second. A flow from one queue q_i to another q_j is considered active when a traffic signal is in a given phase. The sequence of transitions from one phase to another forms a cycle, which in our problem is determined by the domain modeler. There is only one legal ordering of phases however the time spent in each phase may vary within a given minimum and maximum number of seconds. Similarly, queues have a maximum capacity (minimum capacity is 0) also given by the domain modeler. The total number of passenger units moving into a queue from any source may never exceed the queue's maximum capacity. Phases may control multiple flows simultaneously. Between phases there is short period where no traffic may flow.

In this paper we are interested in the kind of problem when part of the network is initially overwhelmed — in other words, a subset of queues are initially considered over capacity; the goal is to move traffic through the network in such a way as to reduce the load on these initially overwhelmed queues to within their maximum capacity. A solution to such a problem describes when each traffic signal should activate the next phase in its cycle until all queues are within their capacity.

A planning problem is defined by a set of states, a set of actions, and a transition function that maps an action applicable in a state to a successor state. A solution is a sequence of actions that maps the initial state to the goal state via the transition function.

To model the above traffic problem as a planning problem, we take a macro approach, defining the state of the network in aggregate in terms of queues and phases. A traffic network state is a set of queues, Q, and traffic signals. Each q_i in Q represents the current load on that queue in terms of the number of passenger units on the queue. Each signal represents the current phase the signal is in and how long it has been in that phase. The set of states in the problem is therefore all possible combinations of loads and phase assignments. An action applicable in a particular traffic network state is the set of phase assignments we could change to, consistent with its minimum and maximum allowable phase times. We assume a fixed time step between a state and its successors (10 seconds in the experiments below). The transition function from one state to another records how the phase assignments change at each signal and simulates how each queue changes over the time interval according to the signaled phase and the flow rates specified by the domain model. The flows between queues depend on the capacities of the links in a non-trivial way, and are computed using a flow model described below. The model is constrained to respect the capacity limits on each link (except for those that are initially overloaded). For our purposes, an initial state is a set of queues and phase assignments in which a subset of queues are above their maximum capacity. A goal state is one in which all queues are at or below their maximum capacity. A solution is a sequence of a set of phase assignments that map the initial set of queues and phase assignments to the goal state in which all constraints queue capacities are honored.

Flow Modeling

In our implementation, we compute how queues change from one state to another in our transition function using a linear program (LP). The input into our transition function is a state of the network, made up of a set of queues Q and a set of signals, at time t and a set of phases to apply to the network over an $interval$. It should return the new state of the network, its set of queues and signals, at time $t + interval$. Constants given to the LP are maximum flow rate for each phase at each intersection, maximum capacity of each link, current count on each link, and the time interval. The maximum $rate(q_i, q_j)$ represents a constant defined by the set of phases given as input to the transition function and may vary from network state to the next. Similarly the count of passenger units in each queue in the network at time t, $current(q)$, is a constant given as input to the LP. The maximum capacity of each link, $capacity(q)$, is a constant given in the problem instance and therefore never changes from state to state. $interval$ is constant given as a parameter to our planner representing the resolution of the problem; we model changes to the network in 10 second intervals.

An easy but inaccurate way to model this problem is for each queue q in Q calculate its maximum flow, the current count of passenger units in q $\times$ the flow rate from q to its downstream queues q_j, $current(q) \times flow(q, q_j)$, subtracting the maximum flow from q and adding it to the downstream queue q_j. However this approach does not take into account the effect of shockwaves on the capacity of q. A

shockwave occurs when limits on the capacity of a downstream queue q_j prevent the upstream queue q from moving its maximum flow; downstream queues have become saturated. The effect propagates backward to upstream queues reducing their maximum flow as well. Consequently the above model would overwhelm downstream queues putting the network into an unrealistic state which fails to honor queue capacities. The LP we use in our experiments, however, allows us to model flows more accurately by taking into account shockwaves. We use this model to simulate the effect of each action during execution. We also take advantage of it during planning in the evaluation of possible actions. The set of equations below describe our LP.

In the LP the objective function maximizes the effective flow, modeled as the number of passenger units moving from queue i to queue j, $flow(q_i, q_j)$, aggregated across all of the active flows in the network:

$$\max \sum_{i,j} flow(q_i, q_j)$$

Note that the decision variables $flow(q_i, q_j)$ are real-valued, rather than integer. We had originally designed our transition function to optimize the maximum integer, rather than continuous, count of passenger units for each $flow(q_i, q_j)$ however in the benchmark problem, where 35% of flows had a maximum flow rate below 0.1 passenger units per second, an ILP at a 10 second resolution would preclude considering these flows in constructing a solution — their $flow(q_i, q_j)$ would always be rounded to 0. Using an LP instead allowed the planner when searching to a depth more than 1 step away to consider the cumulative effect of these flows in the future, a $flow(q_i, q_j) > 0$. The period between phase changes when $rate(q_i, q_j) = 0$ is designed to allow vehicles to clear the intersection (McCluskey and Vallati 2017). Alternatively, another way to interpret these real-valued counts is as expected values.

Our first constraint is that all active flows from one link to another link through an intersection over the time interval must respect the constraints of the traffic dynamics given the infrastructure.

$$flow(q_i, q_j) \leq rate(q_i, q_j) \cdot interval$$

Note a $flow(q_i, q_j)$ is considered active when a signal is assigned a phase that activates the flow; its flow rate $rate(q_i, q_j) > 0$ passenger units per second. Inactive flows have a $rate(q_i, q_j) = 0$ therefore an inactive flow is always $flow(q_i, q_j) = 0$

The capacity of a queue governs the effective flow of all connected queues since no link on the network (other than the initially overloaded queues) can ever exceed its capacity. The flow into a queue minus the flow out of a queue can never exceed the queue's maximum capacity:

$$\sum_i flow(q_i, q) - \sum_j flow(q, q_j) + current(q) \leq capacity(q)$$

The effective flow out of a queue can never exceed the load on the queue at beginning of the interval:

$$\sum_j flow(q_i, q_j) \leq current(q_i)$$

Reversing the direction of a flow is never allowed. To reverse direction traffic would need to flow through the appropriate phase.

$$flow(q_i, q_j) \geq 0$$

The resulting maximizing flows found by the MILP solver are then used to update the count of passenger units in each queue in the network at time $t + interval$. Subtracting the flow from q and adding it to the downstream queue q_j will never push a downstream queue over capacity.

Real-Time Planning

Heuristic state-space search is one of the most popular approaches to planning. In this approach, states of the system being controlled are modeled as vertices in a large graph, with outgoing arcs from a state representing applicable actions that can be taken. A path through the graph corresponds to a sequence of actions chosen by the planner to take the system from its current state toward a more desirable state or goal. The graph is defined implicitly by a staring state and a transition function that lazily instantiates the successor states of a given state only when necessitated by the exploration of the graph. Many exploration strategies exist, the most famous of which is A* search (Hart, Nilsson, and Raphael 1968), which finds optimal plans but can take enormous amounts of time and memory to do so. A* is considered an off-line planner, because it returns only after a complete optimal plan has been found.

An alternative to off-line optimal planning is on-line real-time planning, in which the planner is guaranteed to return with a fixed pre-specified time bound. The catch is that, because it may not be possible to find a complete plan with the specified time bound, the planner need return only the next action for the system to execute. Each action is executed as it is returned, and then a new round of planning is initiated. In this sense, the plan is constructed incrementally as successive calls to the planner return successive actions. The planning is on-line, in that it is interleaved with action execution and can take newly discovered information into account if necessary. This style of operation is also known as receding horizon control. Well-known real-time search algorithms include LRTA* (Korf 1990) and LSS-LRTA* (Koenig and Sun 2009).

In this short paper, we explore very simple real-time methods based on limited lookahead. Because our transition function allows traffic to leave the network but not to enter, it is very unlikely that the network will ever transition back to a previous state. Thus, the complex learning strategies that are necessary to prevent real-time planners from looping are unnecessary in our domain.

One challenge in formulating traffic signal control as a search problem is the large branching factor. If each of k signals can decide to advance to the next cycle, this yields 2^k possible actions for the system as a whole at each time step. In the experiments below, we try three different approaches for dealing with this large branching factor.

The first approach is a simple one-step lookahead, which we also call 'one step, all signals.' It selects which set of

phase assignments to set the network to in the next 10 seconds by generating a list of up to 2^k possible ways to set the k intersections at the next time step. There are at most 2^k possible ways to change the network in the next 10 seconds: for each traffic signal, we may either keep the signal in the current phase or change it to the next phase in the cycle, assuming that it has been in the current phase longer than the phase's minimum length but not longer than its maximum. In many cases this set is smaller than 2^k but it will never be more than 2^k, thus ensuring real-time performance for fixed k. The best set of assignments is determined using a domain-specific scoring function: the sum of the squared overloads, where the overload of a link is the differences between its current load and its maximum capacity.

The second approach looks more than one step ahead, but only permits at most one signal to change at a time. We call it 'one signal, two steps'. It can still guarantee that we can find our next action within a constant amount of time. Searching out to the next 20 seconds (depth=2) generates at most $(k+1)^2$ possible ways the network might evolve. The planner selects the best next configuration of the network that leads to the best possible future.

Our third approach explores the potential of leveraging planning time to look more than one step ahead while considering all signals. Beam search considers intervals out to two steps in the future but, instead of generating all 2^k possible ways to change the network in step two for all 2^k possible ways to change the network in the first step, it scores the first set and selects only the best ten for further evaluation (eg, a beam width of 10).

In the final approach, called 'dynamic cycle,' we examine what happens if we allow the planner to violate the constraint that the phase cycle of each signal is fixed. The predefined cycles given in the benchmark problem were designed to perform well in normal circumstances however the problem we are interested in might not fit that model. In this approach, for a problem with k intersections each of which has p possible phases, the search was allowed to consider up to $(p+1) \times k$ alternatives to a depth of 20 seconds (depth = 2). Like the one-signal two-step strategy, it considered only one signal at a time but it could set it to any of its phases. A short lookahead allowed it to measure the impact of this change on the future state of the system and select the best next action accordingly. However unlike the simple one-step lookahead strategy it did not try all $((p+1) \times k)^2$ possible futures. Only the phase that moved the entire network toward the goal state better than any other phase change in the first 10 second interval (depth = 1) was considered in the next 10 second interval (depth = 2).

Experimental Results

McCluskey and Vallati (2017) kindly provided us with the PDDL description of their benchmark problem modeling seven intersections ($k = 7$) in Manchester, England, in which three queues are significantly overwhelmed; there are 600 more passenger units on these three queues than their maximum capacity. Each intersection can cycle through up to seven phases ($p = 7$), moving multiple queues in each phase. Flow rates from one queue to another vary between 0.01 and 0.8 passenger units per second, median flow rate is 0.16 passenger units per second and interquartile range is between 0.07 and 0.40 passenger units per second. Limits on phase lengths span between 0 and 95 seconds. Periods between phases, when the flow rate is 0, vary between 0 and 25 seconds, and the maximum capacity of any queue on the network is between 13 and 218. In the problem's initial state few queues are completely empty and none are at capacity except the initially overloaded queues which contain an extra 600 passenger units among them. To gain more confidence in our experimental results, we created 20 variations of this problem by randomly distributing the 600 extra passenger units among the problem's original three queues. In the resulting set of variations in some problems some queues carried a greater percentage of the 600 extra passenger units than others. On average these queues were set to four times their initial capacity.

Approach	Execution Time	Planning Time
Fixed	2845	
Domain independent	1500	
Random	4431 ± 229	
One step, all signals	2080 ± 94	0.5235
One signal, two steps	2177 ± 91	0.2492
Beam	2294 ± 112	4.6559
Dynamic cycle	1138 ± 38	0.8648

Table 1: Total execution time, in simulated seconds, taken by each approach to clear congestion from the benchmark problem. Planning time, in seconds, is per planning iteration.

We built a real-time planner using the three approaches outlined above. It was implemented in Python, using Gurobi as the MILP solver. The problem was formulated as an LP maximizing 70 $flow(q_i, q_j)$ variables subject to 270 constraints. As a simple baseline, we also implemented a random planner that selects a random sequence of phase assignments.

Table 1 presents our results. Execution time is the time in seconds that it would take to execute the solution to the problem, the sequence of phase changes that take the problem from its initial state when part of the network is over capacity to the goal state when no part of the network is over capacity. The first two rows of the table are drawn directly from (McCluskey and Vallati 2017), and show how their domain-independent planning approach surpasses the fixed cycle timings currently in use. Later rows in the table represent our results, and must be compared with the previous results with caution. While we have attempted to replicate the flow dynamics of previous work, we have not verified a direct correspondence. Furthermore, they present results on a single instance, while we run on 20 variations and present mean and 95% confidence intervals. Our results show that all planning-based approaches easily beat random phase assignments. Furthermore, both of our real-time planning approaches surpass the fixed strategy, and approach the performance of the off-line method. In the last line of the table, we see the performance that is achievable when the planner is allowed the flexibility to not only decide when a sig-

nal should change but also to which phase it should change. In this case a real-time approach outperforms all other approaches.

Discussion

These results represent our first exploration of real-time search for this problem. We will have additional results by the time of the workshop. For example, one obvious next step is to explore the trade-off between additional lookahead and plan quality.

Our implementation can also be significantly optimized. Currently, we use a heavyweight MILP solver inside our transition function — a lighter one may give faster results on the relatively small problems generated by our benchmark.

Conclusions

We have presented an approach to traffic signal control using real-time heuristic search. Although preliminary, our results suggest that this approach is competitive with other recent work. Using limited lookahead, it can decide on a good way to change the traffic network to move the entire network from its initially overwhelmed state back to normal. We show that repeating this cycle incrementally builds a plan that is better than the benchmark used in other recent work while promising to better scale to a realistically sized problem.

Acknowledgments

We would like to thank Thomas L. McCluskey, Mauro Valleti, and Andrew Kun.

References

Guilliard, I.; Sanner, S.; Trevizan, F. W.; and Williams, B. C. 2016. Nonhomogeneous Time Mixed Integer Linear Programming Formulation for Traffic Signal Control. *Transportation Research Record: Journal of the Transportation Research Board*.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*.

Kiesel, S.; Burns, E.; and Ruml, W. 2015. Achieving goals quickly using real-time search: Experimental results in video games. *Journal of Artificial Intelligence Research*.

Koenig, S., and Sun, X. 2009. Comparing real-time and incremental heuristic search for real-time situated agents. *Autonomous Agents and Multi-Agent Systems*.

Korf, R. E. 1990. Real-Time Heuristic Search. *Artificial Intelligence*.

McCluskey, T. L., and Vallati, M. 2017. Embedding Automated Planning within Urban Traffic Management Operations. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*.

Sharon, G.; Felner, A.; and Sturtevant, N. R. 1992. Exponential Deepening A* for Real-Time Agent-Centered Search. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Smith, S. F.; Barlow, G. J.; Xie, X.-F.; and Rubinstein, Z. B. 2013. Smart Urban Signal Networks: Initial Application of the SURTRAC Adaptive Traffic Signal Control System. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling*.

Sturtevant, N. R., and Bulitko, V. 2011. Learning Where You Are Going and from Whence You Came: h- and g-Cost Learning in Real-Time Heuristic Search. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.