Dynamic Contracts for Distributed Temporal Logic Control of Traffic Networks

Eric S. Kim, Sadra Sadraddini, Calin Belta, Murat Arcak, Sanjit A. Seshia

***Abstract*— Contract-based design is a method to synthesize distributed control strategies for large-scale networks of dynamically coupled systems. We propose a framework for using dynamic contracts for controlling traffic networks from temporal logic specifications. Given a traffic network, we partition it into a number of smaller sub-networks and compute a collection of assume-guarantee contracts for their dynamical interconnections. A central supervisor chooses the best contracts optimally according to the current state of the system. We use model predictive control (MPC) to find control sequences optimally for each sub-network subject to its contract obligations to and promises from its neighboring sub-networks. Our method is correct-by-design. A case study on a mixed urban-freeway network is presented, where the objective is infinite-time congestion avoidance and temporal requirements on the traffic lights in the urban intersections.**

I. Introduction

Realistic models of traffic networks are complex hybrid systems, due to the switching dynamics arising from the state (e.g., a phase change between the congestion and non-congestion regions) and control inputs (e.g., traffic lights). The goal is to design control strategies where the network avoids the congestion region and satisfies a temporal logic specification, with an additional objective of reducing the total induced delay in the network. Real time computation of optimal control decisions in a centralized manner is not possible beyond very small networks. Therefore, large-scale networks are partitioned into smaller manageable sub-networks, and controls are synthesized in a decentralized or distributed fashion [1], [2]. However, dynamical coupling between components is a challenging issue. It can be shown that decentralized control architectures can lead to congestion [3]. Moreover, distributed control is inherently difficult since the traffic demand of a sub-network depends on the control strategy of neighboring sub-networks. Since most interconnections are two-way, this control design paradigm involves circular reasoning [4].

We use contract-based design [5] to formally synthesize traffic control strategies. Contracts are assume-guarantee protocols [6] that determine the i) promises that a sub-network acquires from its environment and ii) obligations of sub-network to its environment and its future evolution. We use MPC to find control sequences for each sub-network optimally subject to the constraints induced by the contracts and the sub-network's own constraints. Contracts have appeared in the form of controlled invariant sets in formal control synthesis [7]–[9], where an individual subsystem treats the unknown coupling from neighboring subsystems as disturbances and computes a control strategy robust to that uncertainty. The invariant sets bound that uncertainty and satisfaction of a contract is enforced as constraints in each sub-network's MPC problem. These invariant sets are typically computed offline and ignore real-time conditions, giving rise to sub-optimal behavior.

This paper's core contribution is a method to attain a better optimum by changing the contracts in reaction to real-time conditions. Each individual sub-network must first be "mined" for contracts over a range of demand severity levels and different local scenarios. The mined contracts are merged into a finite directed graph that serves as a high-level coordinator. Each node corresponds with the contract constraints imposed on each individual sub-network. Unsafe regions of the graph are scenarios where sub-networks' promises to each other are inconsistent. The coordinator graph contains edges only when a hand-off between contracts maintains recursive feasibility. Our protocol accommodates a rich class of temporal specifications that include safety, reachability, and recurrence properties. A running example of a mixed freeway-urban network is used throughout this paper. Simulations show a reduction in overall delay with dynamic contracts and that recursive feasibility is maintained when contract transitions occur.

The authors were funded in part by NSF Grant CNS-1446145 and CCF-1139138. Authors Kim, Arcak, and Seshia are with the Department of Electrical Engineering and Computer Sciences at the University of California Berkeley. Emails: {eskim, arcak, sseshia}@eecs.berkeley.edu Authors Sadraddini and Belta are with the Department of Mechanical Engineering at Boston University. Emails: {sadra, cbelta}@bu.edu

II. Preliminaries

A discrete time interval $I = [a, b)$ is a contiguous subset of $\mathbb{N}$ where $a, b \in \mathbb{N} \cup \{\infty\}$, $a \leq b$, and $b \notin [a, b)$. A closed discrete interval is given by $[a, b] = [a, b) \cup [b]$. Let $x[a, b)$ and $x[a, b]$ represent a slice of the signal $x[0, \infty)$ along the intervals $[a, b)$ and $[a, b]$ respectively.

A. Network Dynamics

We adopt the model from [9]. A traffic network $\mathcal{N}$ can be represented as a graph with a set of links $\mathcal{L}$ representing roads and nodes $\mathcal{S}$ representing intersections. Each link l has an associated vehicle occupancy at time $t \in \mathbb{N}$ denoted by $x^l[t] \in [0, c^l]$ with a maximum capacity $c^l \geq 0$. The entire network state space is $\mathcal{X} = \prod_{l \in \mathcal{L}} [0, c^l]$.

Link l's set of upstream links is $\mathcal{L}^l_{\text{up}}$ and its set of downstream links is $\mathcal{L}^l_{\text{down}}$. Vehicles always flow from an

upstream link to a downstream link. The maximum flow out of link l is $q^l \in [0, c^l]$. Vehicle flow is controlled via traffic lights and ramp meters. Green and red lights at intersections are represented via a control set $\mathcal{U}^l \in \{0, q^l\}$, with $u^l = 0$ indicating that no vehicles may exit link l (red light). If l is actuated by a ramp meter, then $\mathcal{U}^l \in [0, q^l]$ is a continuous variable. If a link is uncontrolled, we have $\mathcal{U}^l = \{q^l\}$. The control input space is denoted by $\mathcal{U} = \prod_{l \in \mathcal{L}} \mathcal{U}^l$. A non-ordered pair of links (l, k) is *antagonistic* if $u^l > 0 \Rightarrow u^k = 0$. The set of all antagonistic pairs is denoted by $\mathcal{A} \subset \mathcal{L} \times \mathcal{L}$.

Flow allocation is determined by functions specifying turning ratios $\beta : \mathcal{L} \times \mathcal{L} \mapsto [0, 1]$ and supply ratios $\alpha : \mathcal{L} \times \mathcal{L} \mapsto [0, 1]$. The turning ratio $\beta(l, m)$ represents the proportion of vehicles exiting l and entering m, is nonzero only for $m \in \mathcal{L}^l_{\text{down}}$ and the ratio sum cannot exceed one, i.e. $\sum_{m \in \mathcal{L}^l_{\text{down}}} \beta(l, m) \leq 1$. The supply ratio $\alpha(k, l)$ represents the proportion of free space in l allocated to upstream link k, and is nonzero only for $k \in \mathcal{L}^l_{\text{up}}$.

A link l's output y^l represents the total vehicles it would like to send and is given by:

$$y^l[t] = \min(x^l[t], q^l, u^l[t]), \tag{1}$$

where the first two terms in the minimization represent the number of vehicles that want to exit link l and the third term controls the output demand. The flow exiting link l is

$$f^l[t] = \min\left(y^l[t], \frac{\alpha(k, l)}{\beta(k, l)}(c^k - x^k[t])\right), \tag{2}$$

where the second term limits flow due to lack of supply downstream. The state update equation is driven by conservation of mass:

$$\begin{aligned} x^l[t+1] = \quad & \min\left(c^l, x^l[t] + \textstyle\sum_{m \in \mathcal{L}^l_{\text{up}}} \beta(m, l) f^m[t]\right. \\ & \left. - f^l[t] + d^l[t]\right), \end{aligned} \tag{3}$$

where $d^l[t]$ is the exogenous demand entering link l at time t. We compactly represent the dynamics above with

$$x[t+1] = F(x[t], u[t], d[t]) \tag{4}$$

$$y[t] = g(x[t], u[t]). \tag{5}$$

The congestion free region of a network is defined as the set $\psi \subseteq \mathcal{X} \times \mathcal{U}$ where for all $l \in \mathcal{L}$ the second term is not a unique minimizer in (2) and c^l is not a unique minimizer of (3). The freeflow condition is a local condition; that is, link l's membership in the freeflow region is determined only by adjacent links. Given the joint space $\mathcal{X} \times \mathcal{U}$ we define a partial order $\leq_{\mathcal{X} \times \mathcal{U}}$. Two points $(x, u), (x', u')$ in this space satisfy $(x, u) \leq_{\mathcal{X} \times \mathcal{U}} (x', u')$ if and only if $x \leq x'$ and $u \leq u'$ coordinate-wise.

Definition 1: A subset $\mathcal{K} \subseteq \mathcal{P}$ is a lower set if for all $q, r \in \mathcal{P}$, $r \in \mathcal{K}$ and $q \leq_{\mathcal{P}} r$ imply $q \in \mathcal{K}$.

The congestion free region ψ is a lower set of $\mathcal{X} \times \mathcal{U}$ and exhibits monotone system dynamics [9].

B. Metric Temporal Logic

Temporal logics generalize Boolean logic to include temporal operators [10], allowing one to make statements with timing dependencies. A proposition is a statement that is either true or false at a given time step, such as "the number of vehicles in link 1 is less than 20" and can be interpreted as a subset of the state-input space $\mathcal{X} \times \mathcal{U}$. Boolean operators $\neg$ (negation), $\wedge$ (intersection/conjunction), and $\vee$ (union/disjunction) are used to make new statements.

Specifications can be viewed as a subset of an appropriate space of signals. Metric temporal logic [11] specifications include Boolean connectives and temporal operators temporal operators associated with a given time interval: $\mathbf{G}_{[a,b)}\phi$ (ϕ holds for *all* time $t \in [a, b)$), $\mathbf{F}_{[a,b)}\phi$ (ϕ holds for *some* time $t \in [a, b)$), $\phi_1 \mathbf{U}_{[a,b)} \phi_2$ (Specification ϕ_2 holds at some time $t \in [a, b)$ and ϕ_1 holds for all time $[a, t)$). Specifications can be composed with one another to express more complex requirements. For instance, $\mathbf{G}_{[0,5)}\mathbf{F}_{[0,3)}\psi \leq 1$ is a recurrence property where for every $t \in [0, 5)$, ψ is satisfied at time $t, t+1$ or $t+2$.

A specification's horizon $h(\phi)$ is the length of a suffix required to determine satisfaction of ϕ. For example, $h(\mathbf{G}_{[0,3)}\psi) = 3$ if ψ is a predicate. The satisfaction of ϕ by signal $s[\cdot]$ at time t is decided by $s[t, t + h(\phi))$; satisfaction at time t is independent of $s[t']$ for $t' \notin [t, t + h(\phi))$.

III. Problem Statement and Approach

Problem Formulation

We partition a traffic network into N smaller sub-networks $\mathcal{N}^i, i = 1, \cdots, N$. A guideline for partitioning traffic networks while taking into account formal specifications was introduced in [9]. We consider traffic specifications of the following form.

$$\phi := \mathbf{G}_{[0,\infty)} \bigwedge_{i=1}^{N} \left(\psi^i \wedge \mathbf{F}_{[0,\infty)} \phi^i\right), \tag{6}$$

where ψ^i is the congestion-free region of network $\mathcal{N}^i$, and ϕ^i is a bounded-time MTL formula describing internal requirements of network $\mathcal{N}^i$.

Problem 1 (Control Synthesis): Given a network $\mathcal{N}$, an initial condition $x[0]$ and a specification ϕ in the form (6), find a control strategy such that ϕ is satisfied.
An optimality criterion is added in Section IV-A.

Motivating Example

Consider a network $\mathcal{N}$ depicted in Fig. 1 which consists of sub-networks $\mathcal{N}^i, i = 1, \cdots, 4$. Sub-network $\mathcal{N}^4$ represents a high capacity freeway while the others are urban areas. All sub-networks are interconnected via on(off)-ramps or urban roads. At each urban intersection, 50% of the vehicles proceed straight, 20% turn left, and 30% turns into an unmodeled external environment (e.g., parking). We have $q^l = 15, c^l = 40$ for urban roads, $q^l = 60, c^l = 25$ for freeways

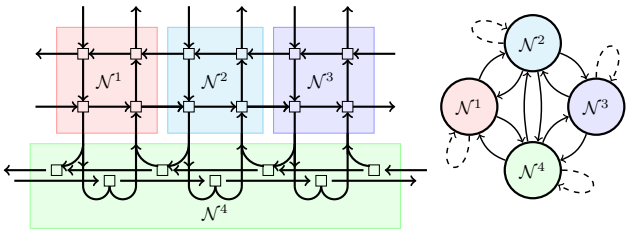

Fig. 1. Example network $\mathcal{N}$ partitioned into four sub-networks $\mathcal{N}^i, i = 1, \cdots, 4$. Contract obligations from Section IV-B are depicted on the right. Solid arrows indicate an obligation from one network to an adjacent sub-network to limit incoming vehicular flow. Dashed arrows represent recursive feasibility obligations from a sub-network to its future self.

and $q^l = 30, c^l = 15$ for ramps. For all entry links to the network, let $d^l[t] \in [0, \frac{1}{4}q^l], t \in \mathbb{N}$ [1].

We aim for multiple control objectives. First, the network must remain in congestion-free region ψ at all times. Second, at each urban intersection traffic lights signalizing vertical and horizontal flows become simultaneously red infinitely often, allowing pedestrians to pass through the intersection in any direction and making the congestion-free specification harder to accomplish.

$$\phi = \mathbf{G}_{[0,\infty)}\Big(\bigwedge_{(l,k)\in\mathcal{A}} \psi \wedge \mathbf{F}_{[0,\infty)}(u_l = 0 \wedge u_k = 0)\Big). \quad (7)$$

This specification is decomposable into the form in (6).

IV. Contract-Based Model Predictive Control

We adopt a distributed MPC approach to solve the controller synthesis problem with temporal logic constraints, as solving the MPC problem for the entire network $\mathcal{N}$ in real time is computationally intractable. A distributed controller synthesis approach is enabled by having each sub-network *concurrently* compute an optimal control trajectory independently [9]. Each sub-network is unaware of adjacent network states during this computation, yet they interact via sending and receiving vehicles to and from adjacent networks. Coordination is enforced by introducing assume-guarantee contracts between networks into each sub-network's MPC problem, ensuring that a sub-network does not inadvertently violate an adjacent sub-network's assumptions and that the distributed control synthesis procedure is sound.

A. Optimization Objective and Constraints

The MPC algorithm executes at each time step t, and aims to minimize the total delay over a horizon T:

$$J = \sum_{\tau=t}^{T+t-1} \sum_{l\in\mathcal{L}} \big(x^l[\tau] - f^l[\tau]\big) \quad (8)$$

by computing a control sequence $u[t, t+T)$ given the current state $x[t]$. Above, $x^l[\tau] - f^l[\tau]$ is the number of vehicles that are forced to remain in l at time τ. Cost (8) is a monotone function with respect to state if the evolution is restricted to the congestion-free region [9].

Both the network dynamics and MTL specification ϕ^i can be encoded as mixed integer constraints. The piecewise affine dynamics are encoded using the scheme in [12] where integer variables are used to detect membership within a set of state space polytopes. Boolean variables that capture predicate satisfaction at different moments in time [13], [14] allow one to encode satisfaction of bounded specification ϕ^i with a finite number of mixed integer constraints, given a sufficiently long MPC horizon.

Assumption 1: Each sub-network $\mathcal{N}^i$ has a common MPC horizon T that is greater than the longest specification horizon. That is, $T > \max_i h(\phi^i)$.

Specification ϕ requires that ϕ^i be satisfied infinitely often. Unlike the optimization procedure which executes with a receding horizon, the constraint ϕ^i can be enforced periodically at times kT where $k \in \mathbb{N}$. Periodic enforcement of ϕ^i for every kT translates to sets of constraints over disjoint intervals $[kT, kT + h(\phi^i))$ each associated with a $k \in \mathbb{N}$. A MPC iteration executing at kT will plan a trajectory over $[k, (k+1)T]$, $k \in \mathbb{N}$ that satisfies ϕ^i. A subsequent MPC iteration executing within that time interval $t \in [kT+1, (k+1)T)$ takes into account a state trajectory over $[kT, t]$ and input trajectories over $[kT, t)$ and can maintain satisfaction of ϕ^i at time kT simply by either i) Executing the input trajectory proposed by the iteration at time kT ii) Computing at time t an input trajectory with lower cost that still satisfies ϕ^i at time kT. While the first option is sufficient to enforce the specification, computing a lower cost trajectory at time t permits the MPC controller to react to incoming vehicles from adjacent sub-networks.

B. Interconnections and Contracts

The distributed MPC scheme contains a circular dependency because each sub-network is unaware of neighboring networks' planned actions. Each sub-network needs to promise neighboring sub-networks that they will satisfy each other's assumptions, but the feasibility of such a promise depends on the actions of one's neighbors. Assume-guarantee contracts are MPC constraints that break this dependency.

Definition 2 (Assume-Guarantee Contract): Network $\mathcal{N}^i$'s assume-guarantee contract $\mathcal{C}^i$ along time interval $[kT, (k+1)T]$ consists of

- Assumption $\phi_a^i(x_*^i[kT], d_*^i[kT, (k+1)T))$ on the incoming demand and vehicles initially in the network:

$$\bigwedge_{l\in\mathcal{L}^i} \big(x^l[kT] \le x_*^l[kT]\big) \quad (9)$$

$$\wedge \bigwedge_{t=kT}^{(k+1)T-1} \Big(\bigwedge_{l\in\mathcal{L}_{\text{in}}^i} \big(d^l[t] \le d_*^l[t]\big) \Big) \quad (10)$$

- Guarantee $\phi_g^i(x_*^i[kT+1, (k+1)T], y_*^i[kT+1, (k+$

[1]The full, detailed, parameter valuations of the network, code and simulation results of this paper are publicly available in http://blogs.bu.edu/sadra/research/dynamic-contracts.

$1)T]$) on the terminal state and output trajectory:

$$\bigwedge_{l\in\mathcal{L}^i}\big(x^l[(k+1)T]\le x^l_*[(k+1)T]\big) \tag{11}$$

$$\wedge\bigwedge_{t=kT}^{(k+1)T}\Big(\bigwedge_{l\in\mathcal{L}^i_{\text{out}}}\big(y^l[t]\le y^l_*[t]\big)\Big) \tag{12}$$

The contract $\mathcal{C}^i$ is characterized by a set of parameters: the initial state $x_*[kT]$, external demand $d_*[kT,(k+1)T)$, terminal state $x_*[(k+1)T]$, and output trajectory $y_*[kT,(k+1)T])$ over output links. Sub-network $\mathcal{N}^i$'s assumption component states conditions over local and incoming links $\mathcal{L}^i$ and $\mathcal{L}^i_{\text{in}}$. The output guarantee is viewed as a signal $y_*[kT+1,(k+1)T])$ that upper bounds output trajectories of $y[kT+1,(k+1)T])$ on links in $\mathcal{L}^i_{\text{out}}$.

Definition 3 (Contract Satisfaction): A sub-network satisfies an assume-guarantee contract at time kT if for all $x[kT]$ and $d[kT,(k+1)T)$ satisfying (9) and (10) respectively, a control sequence $u[kT,(k+1)T)$ exists such that $\mathcal{N}^i$ remains in the local freeflow region ψ^i, and both the guarantee ϕ^i_g and MTL requirement ϕ^i are satisfied at kT.

Contract satisfaction for *all* assumption satisfying scenarios $x[kT]$ and $d[kT,(k+1)T)$ is difficult to certify for general non-linear dynamics. However, within the congestion free region ψ the dynamics exhibit a monotonicity property, where a partial ordering with respect to state trajectories is preserved, and the initial state $x_*[kT]$ and demand $d^i_*[kT,(k+1)T)$ jointly yield the most adversarial environment. An environment that satisfies ϕ^i_a cannot violate the guarantee if $x_*[kT]$ and $d^i_*[kT,(k+1)T)$ doesn't violate the assumption [15]. Thus, synthesizing a satisfying control sequence $u^i_*[kT,(k+1)T)$ for environmental scenario $x_*[kT]$ and $d_*[kT,(k+1)T)$ such that the system satisfies the guarantees and remains congestion free also ensures that the control sequence will be satisfactory under more benign scenarios [16]. A set of contracts is consistent if all sub-network assumptions are implied by the guarantees.

Definition 4 (Assume-Guarantee Parameter Consistency): A set of contracts $\mathcal{C}^1,\dots,\mathcal{C}^4$ are *consistent* if if for all $\mathcal{N}^i$, input links $l\in\mathcal{L}^i_{\text{in}}$ and times $t\in[kT,(k+1)T)$

$$\sum_{k\in\mathcal{L}^l_{\text{up}}}\beta(l,k)y^k_*[t]\le d^l_*[t]. \tag{13}$$

C. Recursive Feasibility with Fixed Contracts

Recursive feasibility can be viewed as a network making a promise to its future self that all future constraints will remain feasible. Recursive feasibility has two components, corresponding to the specification ϕ^i_g constraint and the contract constraint (12). Feasibility of the ϕ^i_g constraint at the kT-th time step has already been established via the initial state condition (9).

Let contract $\mathcal{C}^i$ is be periodically every T steps with fixed parameters. Consider two different MPC executions at times kT and $(k+1)T$. The guarantee constraint ϕ^i_g along the interval $[(k+1)T,(k+2)T)$ is feasible when (9) is satisfied at time $(k+1)T$. The MPC algorithm executing at time kT imposes the terminal state guarantee $x[(k+1)T]\le x_*[(k+1)T]$, which implies that the initial state assumption at $(k+1)T$ with identical contract $\mathcal{C}^i$ is satisfied if:

$$\bigwedge_{l\in\mathcal{L}^i}x^l_*[(k+1)T]\le x^l_*[(k+1)T]. \tag{14}$$

If $\mathcal{C}^i$ satisfies (14) then it is said to be a recursively feasible contract. The final MPC problem for each sub-network consists of the following constraints:

Problem 2 (Distributed MPC): Under the assumption that input demand satisfies (10), Each sub-network $\mathcal{N}^i$ computes a local control sequence $u[kT,(k+1)T)$:

$$\begin{aligned}\underset{u[kT,(k+1)T)}{\text{argmin}}\quad&\sum_{t=kT}^{(k+1)T}\sum_{l\in\mathcal{L}}\big(x^l[t]-f^l[t]\big)\\ \text{s.t.}\quad&(x[kT,(k+1)T],u[kT,(k+1)T))\models\phi^i\\ &\text{Guarantee (12) to adjacent networks}\\ &\text{Terminal State (11), Dynamics constraint (4)}\end{aligned}$$

Assumption ϕ^i_a is encoded in the constraint. (4).

Proposition 1 (Infinite Horizon Spec. Satisfaction): If each network $\mathcal{N}^i$ satisfies its assume-guarantee contract, each assume-guarantee contract $\mathcal{C}^i$ is recursively feasible, and the global initial state $x[0]$ satisfies each initial state (9) assumption, then the distributed MPC algorithm satisfies the global specification (6).

V. Dynamic Contracts

Definition 2 introduced contracts that are uniquely parametrized by $x_*[kT]$, $d_*[kT,(k+1)T)$, $x_*[(k+1)T]$, and $y_*[kT,(k+1)T)$, which do not change over many MPC horizons. Fixed contract parameters may lead to conservative guarantees if the network experiences less demand than expected and $d^l[kT,(k+1)T)\ll d^l_*[kT,(k+1)T)$ elementwise in (10). because conservative assumptions prevent aggressive responses to benign real-time conditions.

In general, a sub-network $\mathcal{N}^i$ can satisfy a collection of m^i assume-guarantee contracts,

$$\mathcal{P}^i=\{\mathcal{C}^i(p^i_1),\dots,\mathcal{C}^i(p^i_{m_i})\}. \tag{15}$$

each associated with different parameters (attributes)

$$p^i[kT,(k+1)T]=\Big(x^i_*[kT],d^i_*[kT,(k+1)T),\\ x^i_*[(k+1)T],y^i_*[kT,(k+1)T)u^i_*[kT,(k+1)T),J^i_*\Big)$$

where p^i is used for notational compactness in (15). Section VI provides a method to generate such a collection. Optimal control sequence $u^i_*[kT,(k+1)T)$ and an induced delay J^i_* are also computed and stored during the contract generation process.

A. Designing a Contract Coordinator

Preserveing formal guarantees restricts how contract parameters may change at runtime. First, contracts parameters must always be consistent in the sense of Definition 4. Second, the notion of recursive feasibility needs to be modified to accommodate a changing set of requirements.

Concretely, a contract coordinator is a transition system with state space $\mathcal{P}=\prod_{i=1}^N\mathcal{P}_i=\prod_{i=1}^N\{\mathcal{C}^i(p^i_1),\dots,\mathcal{C}^i(p^i_{m_i})\}$ designed to ensure that

these two properties are satisfied. Every coordinator transition corresponds to a potential change in contract parameters for each network and may execute every T time steps. After a transition, each sub-network is notified of the contract it must satisfy.

1) Consistent Contract Parameters: A coordinator state $p \in \mathcal{P}$ is a tuple $(p^1_{k_1}, \dots, p^N_{k_N})$ where each network $\mathcal{N}^i$ picks a single contract $\mathcal{C}(p^i_{k_i})$ it would like to satisfy. Not all elements of this space satisfy the contract consistency requirement in Definition 4. For instance in Fig. 1 satisfaction of $\mathcal{N}^1$'s assumption is determined by $\mathcal{N}^2$ and $\mathcal{N}^4$'s guarantees. If contract parameters are such that $\mathcal{N}^2, \mathcal{N}^4$'s guarantees do not jointly imply assumption $\mathcal{N}^1$'s assumption, then these contract parameters are inconsistent.

A subset $\mathcal{P}_v \subseteq \mathcal{P}$ of the parameter space that corresponds to all consistent network-wide parameters. Set $\mathcal{P}_v$ is enumerated via a depth first traversal over a tree with depth N and branching factors m_i. The traversal aggressively prunes branches of the contract space as soon as a contract inconsistency parameter is identified.

2) Contract Transitions and Recursive Feasibility: A contract transition is valid when each sub-network can promise its future self the ability to satisfy the new contract via a transition from the old contract. Given two consistent contract parameters $p, \hat{p} \in \mathcal{P}_v$ where p is for use over $[kT, (k+1)T]$ and $\hat{p}$ is for use over $[(k+1)T, (k+2)T]$, a switch from p to $\hat{p}$ is *valid* if the following element-wise inequality holds:

$$\bigwedge_{l \in \mathcal{L}} x^l_*[(k+1)T] \leq \hat{x}^l_*[(k+1)T]. \tag{16}$$

The left side is the terminal state guarantee from p and the right is the initial state assumption of $\hat{p}$. Recursive feasibility as in Section IV-C is a special case when $p = \hat{p}$.

We define the contract parameters *Viable Graph* (VG) as a directed graph $(\mathcal{P}_v, \mathcal{E}_v)$, where $\mathcal{P}_v$ is the set of nodes, and $\mathcal{E}_v \subseteq \mathcal{P}_v \times \mathcal{P}_v$ is the set of edges such that $\forall (p, \hat{p}) \in \mathcal{E}_v$, the switch from p to $\hat{p}$ is valid. We denote $\mathcal{E}^p_v = \{(p, \hat{p}) | (p, \hat{p}) \in \mathcal{E}_v\}$. A node p is a dead-end if $\mathcal{E}^p_v = \emptyset$. If no dead-end is reached, then there always exists a consistent contract with feasible transition options to other contracts, which by construction implies the following statement.

Proposition 2: Given a infinite-time contract parameter sequence $p_0, p_1, \cdots$, where p_k is used for control synthesis in the time interval $[kT, (k+1)T]$, the specification (6) is satisfied if $(p_k, p_{k+1}) \in \mathcal{E}_v$ and $p_k \in \mathcal{P}_v, \forall k \in \mathbb{N}$.

B. Optimal Contract Coordination

The recursive feasibility property and contract consistency require that no dead-end node in VG is reached. By recursively removing the dead-end nodes and the edges leading to them, we obtain a fixed point which characterizes the *viable kernel graph* (VKG) $(\mathcal{P}_{v,\kappa}, \mathcal{T}_{v,\kappa})$, where $\mathcal{P}_{v,\kappa} \subset \mathcal{P}_v$ and $\mathcal{T}_{v,\kappa} \subseteq \mathcal{P}_{v,\kappa} \times \mathcal{P}_{v,\kappa}$ with the extra property that $\forall p \in \mathcal{P}_{v,\kappa}, \mathcal{E}^p_{v,\kappa} \neq \emptyset$. Once a parameter contract of a node in VKG is chosen, there always exist a feasible handover of the contract to another node in VKG, establishing infinite-time recursive feasibility and consistency.

Each contract p^i corresponds to a cost J^i_* for network $\mathcal{N}^i$, which is the delay induced if control sequence $u^i_*[kT, (k+1)T)$ is applied starting from $x^i_*[kT]$ under the demand assumptions $d^i_*[kT, (k+1)T)$. It follows from monotonicity properties that J^i_* is a upper-bound for possible costs in real-time implementation. Given a contract parameter $p = (p^1_{k_1}, \cdots, p^N_{k_N})$, the sum of associated contract costs is

$$c(p) := \sum_{i=1}^{N} J^i_*(p_{k_i}). \tag{17}$$

Now we determine which contract parameter from VKG nodes to choose at each time $kT, k \in \mathbb{N}$. In order to aim for optimality, we choose the contract parameter for which the infinite-horizon cost $c_\infty(p) = \sum_{k=0}^{\infty} \alpha^k c(p_k)$ is minimal, where $p = p_0$, and $\alpha \in (0, 1)$ is a discount factor to make the cost properly defined. Denote the optimum cost to go from p by c^*_∞ which follows from Bellman's equation [17]:

$$c^*_\infty(p) = c(p) + \alpha \min_{p' \in \mathcal{E}^p_{v,\kappa}} (J^*_\infty(p')) \tag{18}$$

In order to find $c^*_\infty(p), p \in \mathcal{P}_{v,\kappa}$, (18) is cast as a linear program. Finally, at time kT we choose the optimal contract parameter $p^* \in \mathcal{P}_{x[KT]}$ with minimum $c^*_\infty(p)$.

VI. Contract Mining

A delicate tradeoff exists between conservative assumptions which can accommodate an influx of vehicles and aggressive guarantees which quickly dissipate vehicles in the network. We present a heuristic to generate a set of assume-guarantee pairs for each sub-network.

Given a fixed assumption and $\mathcal{N}^i$, a miner is a bounded horizon optimization algorithm that computes a control trajectory that induces minimal guarantees. Guarantee parameters $x^i_*[(k+1)T], y^i_*[kT, (k+1)T)$ are minimal if contract satisfaction as in Definition 3 is infeasible for any smaller guarantee pair such that $\hat{x}^i_*[kT] \leq x^i_*[kT]$ and $\hat{y}^i_*[kT, (k+1)T) \leq y^i_*[kT, (k+1)T)$ with element-wise inequality. The miner's optimization objective can be any monotone function of $\mathcal{X} \times \mathcal{U}$; we opt to minimize a combination of the l_1 and l_∞ norms. After mining, a guarantee is propagated into an assumption for adjacent networks via (13) with the equality is replaced with an inequality, and the mining continues.

Algorithm 1 provides pseudocode which generates `MaxIter` guarantees for every sub-network. The contract sets are initially empty. Infeasibility of the mining algorithm triggers a multiplicative decrease in the initial conditions and disturbance by a factor $\gamma \in (0, 1)$ until the contract is satisfied. Propagation can be visualized over the bottom of Fig. 1 where inter-network promises (solid edges) and a network's recursive feasibility constraint (dashed edges) are both updated. Both contract parameters and the control sequence are saved.

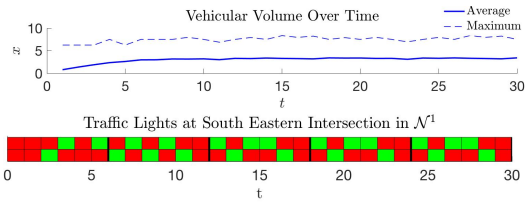

Fig. 2. Simulation Results: [Top] State over Time. [Bottom] The traffic lights history for the links in South-Eastern Intersection of $\mathcal{N}^1$. The lower color is for the light corresponding to the link in North-South direction and the upper one stands for the link in East-West direction. The pedestrian liveness requirement (both lights simultaneously getting red) is satisfied in each round of contract transitions (shown by thick vertical block lines).

Algorithm 1 Guarantee Mining Algorithm

Set N = Number of Sub-networks
for all $i = 1, \ldots,$ `MaxIter` **do**
  for all $i \in \{1, \ldots, N\}$ **do**
    while `True` **do**
      `Feas`, $x[\cdot]$, $u[\cdot]$, $y[\cdot]$= `mine`($\mathcal{N}^i$,$x[0]$, $d[\cdot]$)
      if `Feas` **then**
        `Break`
      $x[0] := \gamma x[0]$, $d[\cdot] := \gamma d[\cdot]$
    `Add to` $\mathcal{N}^i$`s contract set`
    `Propagate Guarantee`

TABLE I

ACCUMULATED DELAYS FOR DIFFERENT CONTROL METHODS

Networks Experiencing Full Demand	Dynamic Contracts Fixed Control	Fixed Contracts MPC	Dynamic Contracts MPC
All	784	753	747
(1,4)	354	381	346
(2,4)	248	244	226
(1,2,4)	513	513	495
(1,2,3)	670	646	635
(1,2)	405	398	394
(1,3)	498	507	460
(1)	238	249	229
(2)	154	122	104
(4)	115	86	85

VII. EXAMPLE

Fig. 1's network is used to evaluate the efficacy of the dynamic contracts system. Algorithm 1 is used to generate a set of 25 contract parameters for each sub-network. There are $|\mathcal{P}_v| = 1363$ consistent contract parameters, of which 664 were members of the viability kernel $\mathcal{P}_{v,\kappa}$. All optimization problems were posed and solved using `Gurobi`'s mixed integer linear program solver [18]. We used the method in Section V to change contract parameters as a feedback of system state every T time steps. We simulated the network for 30 time steps (5 rounds of contract transitions). The satisfaction of the specification was implicitly implied by the fact that the MPC optimization problem was feasible at all times. Sample results are illustrated in Fig. 2.

Table I shows the accumulated delay for different network conditions and control architectures. The first column shows the subset of networks that experience a fully adversarial demand, e.g. the $(1, 3)$ column means that $\mathcal{N}^1, \mathcal{N}^3$ experience the maximum number of incoming vehicles and $\mathcal{N}^2, \mathcal{N}^4$ experience no exogenous demand. As expected, the cumulative delay decreases when the network load decreases. The fixed controller executes a control sequence without state feedback in the interval $(kT, (k + 1)T)$, but permits contract switches every T steps. The MPC controller with fixed contracts achieves similar objective values, but is not strictly better than the fixed control with dynamic contracts, suggesting that contract constraints are a major impediment for achieving a lower delay. The dynamic contracts with MPC column outperforms both other control strategies. The performance gain is greater when the exogenous demand has an asymmetric profile, when dynamic contracts assign higher priority to sub-networks experiencing higher demand.

REFERENCES

[1] E. Camponogara and H. F. Scherer, "Distributed optimization for model predictive control of linear dynamic networks with control-input and output constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 1, pp. 233–242, 2011.

[2] L. B. De Oliveira and E. Camponogara, "Multi-agent model predictive control of signaling split in urban traffic networks," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 1, pp. 120–139, 2010.

[3] J. Gregoire, X. Qian, E. Frazzoli, A. De La Fortelle, and T. Wongpiromsarn, "Capacity-aware backpressure traffic signal control," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 164–173, 2015.

[4] J. Haddad, M. Ramezani, and N. Geroliminis, "Cooperative traffic control of a mixed network with two urban regions and a freeway," *Transportation Research Part B: Methodological*, vol. 54, pp. 17–36, 2013.

[5] B. Meyer, "Applying'design by contract'," *Computer*, vol. 25, no. 10, pp. 40–51, 1992.

[6] K. Chatterjee and T. A. Henzinger, "Assume-guarantee synthesis," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2007, pp. 261–275.

[7] P. Nilsson and N. Ozay, "Synthesis of separable controlled invariant sets for modular local control design," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 5656–5663.

[8] E. S. Kim, M. Arcak, and S. A. Seshia, "Compositional controller synthesis for vehicular traffic networks," in *Decision and Control (CDC), 2015 IEEE Conference on*. IEEE, 2015, pp. 6165–6171.

[9] S. Sadraddini, J. Rudan, and C. Belta, "Formal synthesis for distributed optimal traffic control policies," in *Proceedings of the 8th ACM/IEEE International Conference on Cyber-Physical Systems*. ACM, 2017.

[10] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE, 1977, pp. 46–57.

[11] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

[12] D. Q. Mayne and S. Raković, "Model predictive control of constrained piecewise affine discrete-time systems," *International Journal of Robust and Nonlinear Control*, vol. 13, no. 3-4, pp. 261–279, 2003.

[13] E. M. Wolff and R. M. Murray, "Optimal control of nonlinear systems with temporal logic specifications," in *Robotics Research*. Springer, 2016, pp. 21–37.

[14] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Decision and Control (CDC), 2014 IEEE Conference on*. IEEE, 2014, pp. 81–87.

[15] E. S. Kim, M. Arcak, and S. A. Seshia, "Directed specifications and assumption mining for monotone dynamical systems," in *19th International Conference on Hybrid Systems: Computation and Control*. ACM, 2016, pp. 21–30.

[16] S. Sadraddini and C. Belta, "Safety control of monotone systems with bounded uncertainties," in *55th IEEE Conference on Decision and Control, CDC 2016, Las Vegas, NV, USA, December 12-14, 2016*, 2016, pp. 4874–4879.

[17] D. P. Bertsekas, *Dynamic programming and optimal control*, Vol. 1, No. 2, Belmont, MA, Athena scientific, 1995.

[18] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com