Decision-Making Policies for Heterogeneous Autonomous Multi-Agent Systems with Safety Constraints

Ruohan Zhang[1], **Yue Yu**[2], **Mahmoud El Chamie**[2], **Behçet Açıkmeşe**[2], **and Dana H. Ballard**[1]
[1]Department of Computer Science, University of Texas at Austin, Austin, United States
[2]Department of Aeronautics and Astronautics, University of Washington, Seattle, United States
[1]zharu@utexas.edu, dana@cs.utexas.edu, [2]{yueyu, melchami, behcet}@uw.edu

Abstract

This paper studies a decision-making problem for heterogeneous multi-agent systems with safety density constraints. An individual agent's decision-making problem is modeled by the standard Markov Decision Process (MDP) formulation. However, an important special case occurs when the MDP states may have limited capacities, hence upper bounds on the expected number of agents in each state are imposed. We refer to these upper bound constraints as "safety" constraints. If agents follow unconstrained policies (policies that do not impose the safety constraints), the safety constraints might be violated. In this paper, we devise algorithms that provide safe decision-making policies. The set of safe decision policies can be shown to be convex, and hence the policy synthesis is tractable via reliable and fast Interior Point Method (IPM) algorithms. We evaluate the effectiveness of proposed algorithms first using a simple MDP, and then using a dynamic traffic assignment problem. The numerical results demonstrate that safe decision-making algorithms in this paper significantly outperform other baselines.

1 Introduction

Autonomous multi-agent systems present many challenges in terms of intelligent decision-making algorithms. A distributed algorithm is often more practical and robust than a centralized algorithm. Agents in a distributed decision-making architecture are often self-interested, i.e., they optimize decisions based on local conditions, with limited consideration of global performance and constraints. Therefore, it is challenging to achieve certain global objectives in a distributed multi-agent system. Here we study a particular type of global constraint: *safety* density/capacity constraints, i.e., the expected number of agents in each state must not violate a prescribed upper bound given for that state. Such a global objective challenges traditional algorithms based on Markov Decision Process (MDP) models, such as reinforcement learning (RL) methods. When some states have high reward but low capacity, self-interested agents are likely to violate capacity constraints by cluttering in these states. The distributed decision-making algorithm should avoid such violations without leveraging on a complex inter-agent communication infrastructure, which is the main challenge addressed here.

We first introduce the Safety Constrained MDP (SC-MDP) problem [Arapostathis *et al.*, 2003; Acikmese *et al.*, 2015; El Chamie *et al.*, 2016] and extend the problem formulation from a single agent case to a heterogeneous multi-agent system case, which is the main contribution of this paper. We propose several algorithms that provide safe decision-making policies to ensure the satisfaction of global safety constraints while maximizing the performance of the overall system. Our second contribution is the successful demonstration of the proposed algorithms for decision-making on a well-known Dynamic Traffic Assignment (DTA) problem [Peeta and Ziliaskopoulos, 2001; Hausknecht *et al.*, 2011]. These decision-making algorithms have the following features: 1). The resulting decision-making policies can be implemented in a fully distributed manner and do not require a centralized controller. 2). Communicating/broadcasting each agent's state is not required but can be used as feedback to enhance performance. 3). The resulting safe decision-making policy is randomized and non-stationary. Randomization is the key to remove centralization and communication: agents do not negotiate about whether to transition to a state or not, but rather make statistically independent decisions. The overall system can collectively achieve the desired prescribed behavior. 4). The policy is the same for each agent. Randomization hence also ensures fairness: no agent has higher priority. 5). Decision-making policies can be obtained via solving Linear Programming (LP) problems, and policy synthesis is therefore numerically tractable.

2 Related Work

Constraints in MDPs have been utilized to handle multiple objectives, where policies are computed to maximize one of the objectives while guaranteeing that the other objective values lie within a desired range [Altman, 1999]. Hard constraints can also be imposed by the underlying physical environment (the process model) [Arapostathis *et al.*, 2003; Hsu *et al.*, 2006; Demir *et al.*, 2014; Acikmese *et al.*, 2015], which must be explicitly accounted for in decision policy synthesis. Unlike unconstrained cases, the constraints in MDPs usually cause the optimal policies to be random-

ized rather than deterministic policies [Nain and Ross, 1986; Altman and Shwartz, 1991]. Some recent applications of constrained MDPs include path planning for robotics [Feyzabadi and Carpin, 2014], flight control [Balachandran and Atkins, 2015], and chance-constrained optimal control [Ono *et al.*, 2013; Blackmore *et al.*, 2010]. [El Chamie *et al.*, 2016] presents an SC-MDP algorithm to address a general class of safety constraints on the state probability density function (*pdf*). This approach guarantees the satisfaction of safety constraints not only asymptotically but also during the transition to the steady-state, which classical approaches that are based on state-action frequencies [Altman, 1999] fail to guarantee.

An important application of the SC-MDP algorithm is to the Dynamic Traffic Assignment (DTA) problem [Peeta and Ziliaskopoulos, 2001], which we choose as our example problem to demonstrate the effectiveness of the proposed algorithms. Earlier research utilizing Markov processes in modeling traffic density evolution and optimal assignment include [Hazelton and Watling, 2004; Watling and Cantarella, 2015]. A survey of using multi-agent reinforcement learning in traffic control can be found at [Bazzan, 2009]. DTA is chosen as the key example problem because it is a real-world problem with many challenging constraints and objectives.

3 Method

3.1 Preliminaries

Consider a finite-horizon MDP for a single agent, defined by the tuple $\{\mathcal{H}, \mathcal{S}, \mathcal{A}, \tau_a, R_t, \gamma\}$, where

- $\mathcal{H} = \{0, \cdots, T\}, T \in \mathbb{N}$ is the time horizon. $t \in \mathcal{H}$ is called a time step.
- $\mathcal{S} = \{1, \cdots, n\}, s_t \in \mathcal{S}$ is the state at time step t.
- $\mathcal{A} = \{1, \cdots, p\}, a_t \in \mathcal{A}$ is the action at time step t.
- $\tau_a \in \mathbb{R}^{n \times n}$ is the transition matrix for action a, $\tau_a(s', s) = \mathrm{P}(s_{t+1} = s' | s_t = s, a_t = a)$.
- $R_t \in \mathbb{R}^{n \times p}$ is the reward matrix at t. $R_t(s, a)$ denotes the immediate reward of taking action a in state s at time t. $R_T \in \mathbb{R}^n$ is the terminal reward vector with $R_T(s)$ being the terminal reward for s.
- $\gamma \in (0, 1]$ is the discount factor.

A non-stationary policy for finite Horizon MDP is defined as a sequence of randomized policy matrices $\pi_{0:T-1} := \{\pi_0, \cdots, \pi_{T-1}\}$, $\pi_t \in \mathbb{R}^{n \times p}$, where $\pi_t(s, a) = \mathrm{P}(a_t = a | s_t = s)$. Let Π be the set of all admissible policy sequences, i.e., $\Pi = \{\pi_{0:T-1} | \pi_t \mathbf{1} = \mathbf{1}, \pi_t \geq 0\}$ where $\mathbf{1}$ is the vector of all ones and $\geq$ is an element-wise inequality.

We now introduce several key concepts to define the SC-MDP problem. Let $\mathbf{x}_t \in \mathbb{R}^n$ denote the state probability density distribution function (*pdf*) at t, where $\mathbf{x}_t(s)$ denotes the *probability of finding the agent in s at t*, i.e., $\mathbf{x}_t(s) = \mathrm{P}(s_t = s)$. Let $M_t \in \mathbb{R}^{n \times n}$ be the Markov chain (MC) transition matrix at t, where:

$$\begin{aligned} M_t(s', s) &= \mathrm{P}[s_{t+1} = s' | s_t = s] \\ &= \textstyle\sum_{a \in \mathcal{A}} \pi_t(s, a) \tau_a(s', s). \end{aligned} \tag{1}$$

Let $\bar{\mathbf{r}}_t(s) = \mathbb{E}[r_t(s, a_t)] = \sum_{a \in \mathcal{A}} \pi_t(s, a) R_t(s, a)$ be the expected immediate reward of s at t given a policy π_t. For $t = T$, let $\bar{\mathbf{r}}_T = R_T$.

Let

$$U_t = \bar{\mathbf{r}} + \gamma M_t^\intercal U_{t+1}, \qquad U_T = \bar{\mathbf{r}}_T \tag{2}$$

where $U_t(s)$ denotes the value of s at t under policy $\pi_{t:T-1}$, i.e., expected accumulated discounted rewards from t to T under policy $\{\pi_t, \cdots, \pi_{T-1}\}$ with $s_t = s$.

For a given policy $\pi_{0:T-1}$, the random process induced by the state of the system at different time steps: $\{s_0, s_1, \ldots, s_T\}$ is a discrete-time Markov chain (DTMC) whose state space is $\mathcal{S}$. The state *pdf* evolves according to the MC transition matrix

$$\mathbf{x}_{t+1} = M_t \mathbf{x}_t, \quad t = 0, \ldots, T-1. \tag{3}$$

Note that M_t is a linear function of π_t as shown in (1).

Performance Metric

We use the expected accumulated discounted rewards as the performance metric for a given policy, at time-step t:

$$v_t^{\pi_{t:T-1}}(\mathbf{x}_t) = \mathbb{E}\left[\sum_{i=t}^{T-1} \gamma^{i-t} R_i(s_i, a_i) + \gamma^{T-t} R_T(s_T)\right] \tag{4}$$

denotes the expected accumulated discounted rewards from t to T under policy $\{\pi_t, \cdots, \pi_{T-1}\}$ where $\mathbf{x}_t$ is the state *pdf* at t. Naturally, the performance metric to evaluate $\{\pi_0, \cdots, \pi_{T-1}\}$ is $v_0^{\pi_{0:T-1}}(\mathbf{x}_0)$. Using the linearity of the expectation operator, (4) becomes:

$$v_t^{\pi_{t:T-1}}(\mathbf{x}_t) = \sum_{i=t}^{T} \gamma^{i-t} \mathbf{x}_i^\intercal \bar{\mathbf{r}}_i = \mathbf{x}_t^\intercal U_t \qquad t \in \mathcal{H} \setminus \{T\} \tag{5}$$

3.2 Safety Constrained MDP Problem

The unconstrained MDP optimal policy π^* is the policy that maximizes the performance metric.

$$\pi^* = \underset{\pi_{0:T-1} \in \Pi}{\operatorname{argmax}}\, v_0^{\pi_{0:T-1}}(\mathbf{x}_0)$$

π^* can be solved using *backward induction* algorithm [Puterman, 1994, p. 92]. Now, we introduce Safety Constrained MDP (SC-MDP) [El Chamie *et al.*, 2016], which is an MDP with the *safety constraints*:

$$\mathbf{x}_t \leq \mathbf{d}, \forall t \in \mathcal{H}, \tag{6}$$

where $\leq$ denotes the element-wise inequalities, and $\mathbf{d} \in [0, 1]^n$ is a vector giving upper bounds on the system state *pdf*. We assume $\mathbf{d}$ is independent of t. Intuitively, at any given t, $\mathbf{d}$ constrains the probability of finding a single agent in each state. Later we will show that for a multi-agent system, $\mathbf{d}$ constrains the density of agents in each state.

To solve the SC-MDP policy is equivalent to solving the following constrained optimization problem:

$$\begin{aligned} \underset{\pi_{0:T-1} \in \Pi}{\text{maximize}} \quad & v_0^{\pi_{0:T-1}}(\mathbf{x}_0) \\ \text{s.t.} \quad & \mathbf{x}_t \leq \mathbf{d}, \quad \forall t \in \mathcal{H} \end{aligned} \tag{7}$$

Note that according to (3), $\mathbf{x}_t = M_{t-1} \dots M_1 M_0 \mathbf{x}_0$, hence although each M_i is linear in the decision variables π_i, constraints (6) are non-convex in general. Also directly applying Dynamic Programming (DP) will not help in this case: if we try to apply *forward induction*, i.e., for $t = 0, \cdots, T-1$

$$\hat{\pi}_t = \underset{\mathbf{x}_t \leq \mathbf{d}}{\text{argmax}_{\pi_t}} \ \mathbf{x}_t^\intercal U_t \tag{8}$$

Then at t-th iteration, according to (2), U_t is unknown since it depends on $\{\pi_t, \cdots, \pi_{T-1}\}$. Similarly, according to (3), $\mathbf{x}_t$ is unknown at t-th iteration of *backward induction*.

3.3 Multi-Agent System

The constraints in (6) give upper bounds on the probability distribution of a single agent. We now extend the single-agent results [El Chamie *et al.*, 2016] to a multi-agent system using the law of large numbers. We will show that having bounds on the probability distribution of being in a state induces bounds on the expected number of agents in that particular state.

Assume now that N agents are operating in the same environment, i.e., each agent uses the same motion decision policy obtained by solving the MDP. The probability distribution of agent k at t, given by $\mathbf{x}_t^{(k)}$, has the following interpretation: $\mathbf{x}_t^{(k)}(s)$ represents the probability of finding the agent k in s at t. Since all N agents have identical MDP,[1] then $\sum_{k=1}^N \mathbf{x}_t^{(k)}(s)$ describes the expected number of agents in s at t. Let $\mathbf{c}_t = [\mathbf{c}_t(1), ..., \mathbf{c}_t(n)]^\intercal$ denote the actual number of agents in each state. Then, $\mathbf{c}_t(s)$ is generally different from $\sum_{k=1}^N \mathbf{x}_t^{(k)}(s)$, although it follows from the i.i.d. agent realizations that

$$\sum_{k=1}^{N} \mathbf{x}_t^{(k)} = \mathbb{E}[\mathbf{c}_t], \tag{9}$$

and from the law of large numbers we will show next that that $\mathbf{c}_t/N \to \mathbf{x}_t$ as $N \to \infty, \forall t \in \mathcal{H}$ where $\mathbf{x}_0 = \sum_{k=1}^N \mathbf{x}_0^{(k)}/N$ and $\mathbf{x}_t$ follows the dynamics of the n-dimensional system (3).

The idea behind extending the probabilistic guidance of a single agent to multi-agents using the law of large numbers is to control the propagation of the probability vector, $\mathbf{x}_t$, rather than each individual agent's position. While, in general, $\mathbf{c}_t/N \neq \mathbf{x}_t$, it will always be equal *on average*, and can be made *arbitrarily close* to $\mathbf{x}_t$ by making the number of agents N sufficiently large [Demir *et al.*, 2015].

Consider a multi-agent system comprised of N agents. Suppose that each agent has the following probability distribution over a given domain at t:

$$\mathbf{x}_t^{(k)}(i) = P(s_t^{(k)} = i), \quad k = 1, ..., N, \quad \forall t \in \mathcal{H} \tag{10}$$

Then the following result holds.

Proposition 1. *Consider a multi-agent system of N agents such that $\boldsymbol{x}_0 := \frac{\sum_{k=1}^N \mathbf{x}_0^{(k)}}{N}$ at time $t = 0$. Further, suppose that each agent acts independently such that $\boldsymbol{x}_{t+1}^{(k)} = M_t(\pi_t)\boldsymbol{x}_t^{(k)}$, $k = 1, ..., N$, where $M_t(\pi_t)$ is a column stochastic Markov matrix and it is defined by the same policy π_t for all agents. Then $\boldsymbol{x}_{t+1} = M_t \boldsymbol{x}_t$, $t = 0, 1, 2, ...$, and*

$$P\left(\lim_{N\to\infty} \frac{\boldsymbol{c}_t}{N} = \boldsymbol{x}_t\right) = 1, \quad \forall t \in \mathcal{H} \tag{11}$$

where $\boldsymbol{c}_t$ is the vector of number of agents in each state at t.

Proof. Next, consider $\mathbf{x}_t(i)$ which is the probability of finding any agent in state i at time t. Consider a new Random Variable (RV), $Z_t^{(k)}(i)$, such that $Z_t^{(k)}(i) = 1$ if agent k is in i at t (i.e., if $X_t^{(k)} = i$), and zero otherwise. Clearly, $\mathbb{E}[Z_t^{(k)}(i)] = \mathbf{x}_t^{(k)}(i)$, where $\mathbb{E}$ is the expectation operator and $Z_t^{(k)}(i)$, $k = 1, ..., N$, form i.i.d. RVs. Then it follows that $\mathbf{c}_t(i) = Z_t^{(1)}(i) + \cdots + Z_t^{N}(i)$. Note that $\mathbf{x}_t(i) = \mathbb{E}[\sum_{k=1}^N Z_t^{(k)}(i)/N]$, and since $Z_t^{(k)}(i)$, $k = 1, ..., N$ are i.i.d. RVs and $\mathbb{E}[|Z_t^{(k)}(i)|] < \infty$, we can use the strong LLN theorem, [Chung, 2001, Theorem 5.4.2], to conclude that:

$$P\left(\lim_{N\to\infty} \frac{\mathbf{c}_t(i)}{N} = \mathbf{x}_t(i)\right) = 1, \tag{12}$$

which then gives the desired result. □

This result shows that when all agents evolve independently according to an identical MC transition matrix, the ensemble of agent positions has a distribution that approaches $\mathbf{x}_t$ with probability one as N increases, which makes constraints on $\mathbf{x}_t$ constrain agent density asymptotically. Hence results for a single agent in [El Chamie *et al.*, 2016] can be applied to a multi-agent system. We refer solving a multi-agent SC-MDP policy as a *policy synthesis* problem.

3.4 Multi-Agent Policy Synthesis Algorithms

First, we define the following two sets:

Definition 1. *Let $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n : 0 \leq \mathbf{x} \leq \mathbf{d},\ \mathbf{1}^\intercal \mathbf{x} = 1\}$ and $\mathcal{C}(\mathcal{X}) = \bigcap_{\mathbf{x}\in\mathcal{X}} \mathcal{C}(\mathbf{x})$ with*

$$\mathcal{C}(\mathbf{x}) = \left\{\pi \in \mathbb{R}^{n\times p} : \pi \mathbf{1} = \mathbf{1},\ \pi \geq 0,\ M(\pi)\mathbf{x} \leq \mathbf{d}\right\}$$

where $M(\pi)$ is defined as in (1)*. We refer to $\mathcal{X}$ as the safe density set, and to $\mathcal{C}(\mathcal{X})$ as the safe policy set for $\mathcal{X}$.*

Note that $\mathcal{C}(\mathbf{x})$ is a convex set of policies because $M(\pi)$ is linear in π, and $\mathcal{C}(\mathcal{X})$ is also convex since it is the intersection of convex sets. The convexity of these sets allows them to be characterized by linear constraints.

As discussed before, DP is not directly applicable to SC-MDP: neither *forward induction* nor *backward induction* will work because either the value of U_t or $\mathbf{x}_t$ is undetermined at t-th iteration. To resolve this dilemma, we develop heuristics that either provide an estimate of U_t to be used for *forward induction* or an estimate of $\mathbf{x}_t$ to be used for *backward induction*: Algorithm 1 uses *forward induction* by estimating U_t with the corresponding value function V_t of unconstrained MDP. It is based on the assumption that $\mathbf{x_0}$ is known. Algorithm 2 [El Chamie *et al.*, 2016] uses *backward induction* by considering worst case $\mathbf{x}_t$ from safe density set $\mathcal{X}$ that minimizes $\mathbf{x}_t^\intercal U_t$ but remains in safe density set $\mathcal{X}$. It does not require knowing $\mathbf{x}_0$ itself, but assumes $\mathbf{x}_0 \in \mathcal{X}$. In this way,

[1] We later introduce heterogeneous agents where agents can have different transition and reward functions.

it only seeks policies in $\mathcal{C}(\mathcal{X})$. Algorithm 3 improves Algorithm 2 via a projection technique, which finds the closest policy to the optimal policy of unconstrained MDP among the safe policies obtained from Algorithm 2. We use the Frobenius norm as a distance measure, but other norms produce similar results. Algorithm 4 improves over Algorithm 2 when $\mathbf{x}_0$ is given via both *forward induction* and *backward induction* by providing estimates to U_t and $\mathbf{x}_t$ in succession. All these algorithms are based on LP and can be implemented using any standard LP solvers. A comparison of these algorithms is shown in Table 1.

Algorithm	x_0 given	$x_0 \in \mathcal{X}$	Policy Property
SCF	✓	-	$\pi_t \in \mathcal{C}(\mathbf{x}_t)$
SCWC	-	✓	$\pi_t \in \mathcal{C}(\mathcal{X})$
SCPRO	-	✓	$\pi_t \in \mathcal{C}(\mathcal{X})$
SCBF	✓	✓	$\pi_t \in \mathcal{C}(\mathcal{X})$

Table 1: Algorithm assumptions and policy properties.

3.5 Heterogeneous Agents

The single class multi-agent model can be extended to a heterogeneous multi-agent system where agents can be categorized into l different classes. Agents' transition model, reward function, and optimal policies remain homogeneous within the same class but heterogeneous in different classes. An example of heterogeneity in DTA is when different size of vehicles contribute differently in traffic congestion. Note that all these heterogeneities can be addressed by simply extending the state space $\mathcal{S}$. However, safety constraints (6) are usually imposed on the common environment shared by all agents and not on the extended state space $\mathcal{S}$. We can handle this heterogeneity using a linear operator $L \in \mathbb{R}^{m\times n}$ where $ml = n$ that maps the agent state space $\mathcal{S}$ to an environment state space where safety upper bounds are imposed. Then (6) can be generalized to:

$$L\mathbf{x}_t \leq \mathbf{d}, \quad \forall t \in \mathcal{H} \tag{13}$$

The algorithms can still be implemented by a slightly modified LP with constraints (13) as they are still linear in $\mathbf{x}_t$.

Algorithm 1 SC-Forward Induction (SCF)

1: Let $V_T = \bar{\mathbf{r}}_T$, for $t = T-1, \cdots, 1, \forall s \in \mathcal{S}$,

$$V_t(s) = \max_{a\in\mathcal{A}} \left[R(s,a) + \gamma \sum_{s'} \tau_a(s',s) V_{t+1}(s) \right]$$

2: Given $\mathbf{x}_0$, for $t = 0, \cdots, T-1$, compute policy and distribution at next time step

$$\hat{\pi}_t = \operatorname*{argmax}_{\pi_t \in \mathcal{C}(\mathbf{x}_t)} (M_t(\pi_t)\mathbf{x}_t)^\intercal V_{t+1}$$

$$\mathbf{x}_{t+1} = M_t(\hat{\pi}_t)\mathbf{x}_t$$

Algorithm 2 SC-Worst Case (SCWC)

1: Set $U_T = \bar{\mathbf{r}}_T$.

2: For $t = T-1, \cdots, 0$, given U_{t+1}, compute the policy

$$\hat{\pi}_t = \operatorname*{argmax}_{\pi_t \in \mathcal{C}(\mathcal{X})} \min_{\mathbf{x}\in\mathcal{X}} \mathbf{x}^\intercal \left(\bar{\mathbf{r}}_t(\pi_t) + \gamma M_t^\intercal(\pi_t) U_{t+1} \right), \tag{14}$$

and

$$U_t = \bar{\mathbf{r}}_t(\hat{\pi}_t) + \gamma M_t^\intercal(\hat{\pi}_t) U_{t+1}.$$

Usually $\hat{\pi}_t$ obtained from step 2 is not unique, let $\tilde{\mathcal{C}}_t(\mathcal{X})$ denote the corresponding set of policies.

Algorithm 3 SC-Projection (SCPRO)

1: Run Algorithm 2 to get $\tilde{\mathcal{C}}(\mathcal{X})_t, \forall t \in \mathcal{H} \setminus \{T\}$

2: Let $V_T = \bar{\mathbf{r}}_T$, for $t = T-1, \cdots, 0$, compute the optimal policy for unconstrained MDP: $\forall s \in \mathcal{S}$

$$a_t^*(s) = \operatorname*{argmax}_{a\in\mathcal{A}} \left[r(s,a) + \gamma \sum_{s'} \tau_a(s',s) V_{t+1}(s) \right]$$

let $\forall s \in \mathcal{S}, \pi_t^*(s, a_t^*(s)) = 1$, and 0 elsewhere, and

$$V_t = \bar{\mathbf{r}}_t(\pi_t^*) + \gamma M_t^\intercal(\pi_t^*) V_{t+1}$$

3: For $t = T-1, \cdots, 0$, compute the policy

$$\hat{\pi}_t = \operatorname*{argmin}_{\pi_t \in \tilde{\mathcal{C}}(\mathcal{X})_t} \| \pi_t - \pi_t^* \| \tag{15}$$

Algorithm 4 SC-Backward-Forward Induction (SCBF)

1: Run Algorithm 2 to get an initial safe policy $\{\pi_0^0, \cdots, \pi_{T-1}^0\}$

2: Let $\hat{\pi}_t = \pi_t^0 \quad \forall t \in \mathcal{H} \setminus \{T\}$

3: **Forward Induction**:
Given $\mathbf{x}_0$, for $t = 0, \cdots, T-1$, compute $\mathbf{x}_{t+1}$ based on $\mathbf{x}_t$ and $\hat{\pi}_t$

$$\mathbf{x}_{t+1} = M_t(\hat{\pi}_t)\mathbf{x}_t$$

4: **Backward Induction**:
Let $U_T = \bar{\mathbf{r}}_T$, for $t = T-1, \cdots, 0$, re-compute the policy based on U_{t+1} and $\mathbf{x}_t$,

$$\hat{\pi}_t = \operatorname*{argmax}_{\pi_t \in \mathcal{C}(\mathcal{X})} \mathbf{x}_t^\intercal \left(\bar{\mathbf{r}}_t(\pi_t) + \gamma M_t^\intercal(\pi_t) U_{t+1} \right)$$

and

$$U_t = \bar{\mathbf{r}}_t(\hat{\pi}_t) + \gamma M_t^\intercal(\hat{\pi}_t) U_{t+1}$$

5: Repeat steps 3 and 4 until $\{\hat{\pi}_0, \cdots, \hat{\pi}_{T-1}\}$ converges.

4 Experiments and Results

4.1 Resource Collection

We first evaluate the algorithms using a simple 11-state MDP, as shown in Fig. 1. 100 agents are placed at home state initially. The other ten states each provides different amount of reward at every time step, with different capacity upper bound. If more agents than the upper bound capacity enter

a state, the extra ones will not receive reward. The agents can stay or move from any state to any other state. An action succeeds with probability .75, and failure causes the agent to stay. Let $C(s_i)$ denote the capacity upper bound of state s_i. Its density upper bound is $d_i = \min\{C(s_i)/N, 1\}$. We set $\gamma = 0.99$.

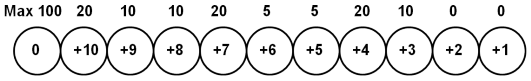

Figure 1: Multi-agent resource collection task.

Results We evaluate the safety and optimality performance. Safety is measured by the total number of times that an agent violates capacity upper bound. Optimality is measured by the total reward accumulated. The results are shown in Fig. 2. Comparison algorithms are centralized, random, safe, and greedy. The centralized algorithm starts from the state with the highest reward and allocates maximum number of agents that state allows, then moves to the state with second highest reward, etc. The random agents choose an action uniformly at random. The greedy agents choose a state with probability proportional to the state's reward. The safe agents choose a state with probability proportional to the state's density upper bound.

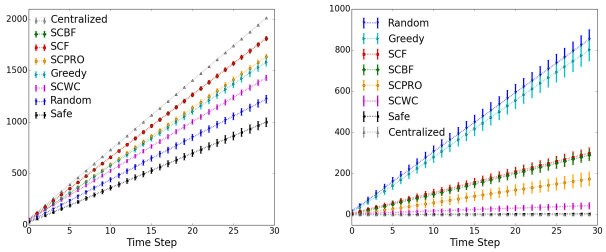

Figure 2: Algorithm performance. Left: accumulated reward. Right: violation count. Error bars: 95% CI for 100 trials.

Overall, in terms of reward, SCBF and SCF perform closely to centralized algorithm. SCPRO and SCWC algorithms are conservative and results in collecting less reward, but they still perform better than or comparable to greedy, random, and safe algorithms. In terms of violation, all SC algorithms have significantly less violations comparing to random and greedy algorithms. SCWC algorithm resulted in the fewest violation count. Additional experiments were performed on other randomly generated density vectors and the conclusions are similar.

This experiment demonstrates one motivation of SC algorithms: it is not always trivial to achieve optimality while maintaining safety in distributed systems. SC algorithms balance optimality and safety better than the baseline distributed algorithms, and have close performance to the optimal centralized algorithm.

4.2 Dynamic Traffic Assignment

We apply our algorithm to a dynamic traffic assignment (DTA) problem in a multi-intersection environment [Hausknecht *et al.*, 2011] as shown in Fig. 3. The task is to direct agents (vehicles) to roads and intersections that have low usage to reduce congestion, while ensuring that all agents arrive at their destinations. For simplicity, all roads are two-way, and U-turns are allowed in all states. One-way roads and traffic rules can be introduced by changing the transition function of the MDP. The roads have different carrying capacities. Violating the capacity upper bound results in congestion modeled by the fundamental diagram [Greenshields *et al.*, 1935]. The diagram maps the current traffic level to the average speed. This effect is simulated by the transition function in the MDP. Agents have two sizes: a large agent occupies $\alpha_1 = 3$ units of capacity, and a small one occupies $\alpha_2 = 1$ unit.

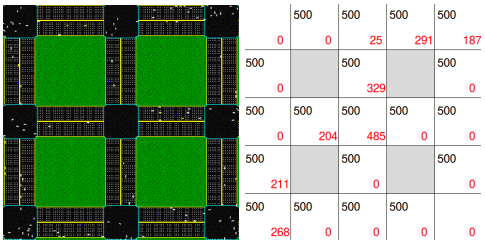

Figure 3: Left: the original dynamic traffic assignment simulator with multiple intersections. Right: discretized version; black numbers indicate state capacity upper bound; red numbers indicate the current number of agents.

We discretized the state space as in Fig. 3, and implemented a discretized version of the Automatic Intersection Manager simulator [Dresner and Stone, 2008]. Note that a two-way road block is a single state, rather than two states for each direction, since [Hausknecht *et al.*, 2011] introduces dynamic lane reversal hence a block's carrying capacity can be shared for both directions. Let N denote the total number of agents. The problem MDP with $\gamma = 0.99$ is defined as:

- $\mathcal{S}$: a state is a tuple $\langle position, destination, type \rangle$. $position$ indicates the current position. $type$ indicates whether an agent is large or small. $destination$ is included to better define the reward function.
- $\mathcal{A}$: {North, South, East, West, Stay}.
- τ: we use the transition function to simulate the speed change according to the fundamental diagram. The slower the speed, the higher the probability that an agent will stay when trying to move from state s to s':

$$P(s'|s,a) = \begin{cases} 0 & s' \text{ is offroad} \\ 1 & s' = s \text{ (Stay)} \\ 1 & c_t(s) \le C(s) \\ e^{-\frac{c_t(s)-C(s)}{C(s)}} & c_t(s) > C(s) \end{cases}$$

- $\mathcal{R}$: an agent receives a reward of +1 at the state in which its $position$ equals to $destination$. As mentioned before, this definition is to allow agents with different objectives to share a single MDP and reward function.
- $L = [\ \alpha_1 I \quad \cdots \quad \alpha_k I\ ]$, where $I \in \mathbb{R}^{n \times n}$ is the identity matrix, $\alpha_i \in \mathbb{R}$ is the size for i-th type of agents.

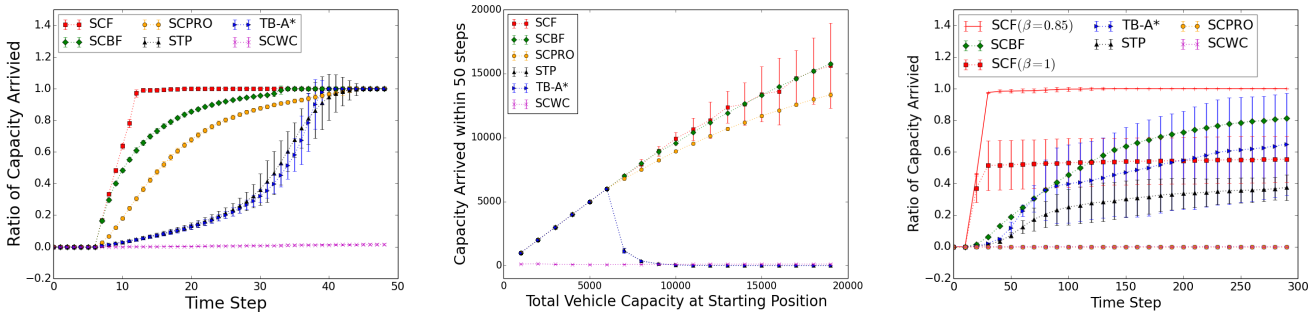

Figure 4: DTA results. Left (small map): ratio of total agent capacity arrived over 50 time steps (6000 capacity). Middle (small map): total agent capacity vs. capacity arrived within 50 time steps. Right (large map): ratio of total agent capacity arrived over 300 steps (48000 capacity). Error bars: 95 % CI for 50 trials.

At the beginning of the simulation, all agents broadcast their locations and destinations, hence each agent can define its SC-MDP problem and calculate the policy.

Results We compare SC algorithms with a shortest path (STP) and a Time-based A* (TB-A*) algorithm [Hausknecht *et al.*, 2011]. An STP agent chooses the shortest path (break tie randomly), without considering other agents. The TB-A* agent estimates its expected shortest-time path given the traffic of every state (hence transition probability). TB-A* requires observation and path re-planning at every time step.

We test two maps. The first one has 6 roads and 9 intersections (Fig. 3). The capacity upper bound for each block is 500. We simulate one traffic flow: agents travel from the upper left corner to the lower right corner. Half of the agents are large agents and the other half are small. The planning horizon for SC algorithms is 50 time steps.

The results are shown in Fig. 4. The leftmost figure shows percentage of total capacities arrived over time steps. 1500 large and 1500 small agents (6000 total capacity) are placed at the starting state. All SCF agents quickly arrive at their destination, while SCBF, SCPRO agents arrive slower but still faster than STP and TB-A*. In runtime simulations, we observe that SC agents wait nicely in line to move through bottlenecks, without a centralized controller nor explicit coordination.

In the middle figure, we show the capacity arrived in 50 time steps while varying the total capacity. When the total capacity is low (< 7500), all algorithms except SCWC can ensure all agents arrive at their destination within 50 time steps. When total capacity is increased, STP and TB-A* suffer from congestion and only few agents arrived. SCF and SCBF both perform well. SCF has larger variance, because SCF agents move faster but are more likely to cause congestion due to their more aggressive randomized policy.

The second map is larger with 8 roads and 16 intersections. We simulate four traffic flows: a quarter of the agents start at each corner and travel to the diagonal corner. Road capacities are sampled from $[800, 1200]$. Results are shown by the rightmost plot in Fig. 4. The performance of SCPRO degrades to be the same as SCWC. Although SCBF and SCF still perform better than STP and TB-A*, the SCF policy is risky and leads to congestion, since it always pushes maximum number of agents towards their destinations. This is problematic with multiple traffic flows. The problem is alleviated by lowering the actual density upper bound in Algorithm 1, namely multiplying the actual bound by a β factor. The resulting policy is less likely to cause congestion, as indicated by the top red curve. When varying total capacity, the result is similar to the one of the small map: SCF and SCBF significantly outperform STP and TB-A* when total capacity grows large.

5 Conclusion and Future Work

We extend the previous SC-MDP framework [El Chamie *et al.*, 2016] to a distributed heterogeneous multi-agent system, where one aims to bound the agent density over a finite horizon. We develop three novel algorithms (SCPRO, SCBF, and SCF) that improve the performance of the original algorithm (SCWC). SCF and SCBF produce the best results in terms of safety and optimality, since additional information (initial distribution of agents) is available. However, since the guarantee is probabilistic, one might consider to be conservative (more safe), i.e., use SCBF or lower the β factor in SCF.

An alternative approach for multi-agent density control could be multi-agent reinforcement learning [Hu *et al.*, 1998]. One can modify the reward function to achieve global objectives, such as taxation or toll [Bnaya *et al.*, 2013; Xiao *et al.*, 2014]. By assigning penalty to the congested states, vehicles can recalculate their policy using the updated reward function. However, it is non-trivial to determine the magnitude of the penalty, especially with rewards for reaching the destination. SC-MDP framework's advantage is that it defines a constrained optimization problem, hence clearly separating the constraints from the optimization objective.

An extension of our algorithm is to enforce a lower bound on agent density as well, i.e., $\underline{\mathbf{d}} \leq L\mathbf{x}_t \leq \bar{\mathbf{d}}$. Notice this can also be written in the form of (13) with proper choice of L. These extensions may be applied to multi-agent problems such as surveillance.

Although the algorithms assume an open-loop policy after the first episode, they can also observe the actual agent distribution $\mathbf{x}_t$ at every episode or every several episodes. The agents can treat $\mathbf{x}_t$ as a new $\mathbf{x}_0$ and recalculate their policies. This requires global observation but not explicit coordination.

Acknowledgments

We would like to thank Peter Stone for helpful comments and suggestions. This research was supported in part by Defense Advanced Research Projects Agency (DARPA) Grant No. D14AP00084 and the National Science Foundation (NSF) under Grant No. CNS-1624328.

References

[Acikmese *et al.*, 2015] Behcet Acikmese, Nazli Demir, and Matthew W Harris. Convex necessary and sufficient conditions for density safety constraints in markov chain synthesis. *Automatic Control, IEEE Transactions on*, 60(10):2813–2818, 2015.

[Altman and Shwartz, 1991] Eitan Altman and Adam Shwartz. Markov decision problems and state-action frequencies. *SIAM Journal on Control and Optimization*, 29(4):786–809, 1991.

[Altman, 1999] E. Altman. *Constrained Markov Decision Processes*. Stochastic Modeling Series. Taylor & Francis, 1999.

[Arapostathis *et al.*, 2003] Aristotle Arapostathis, Ratnesh Kumar, and Sekhar Tangirala. Controlled markov chains with safety upper bound. *Automatic Control, IEEE Transactions on*, 48(7):1230–1234, 2003.

[Balachandran and Atkins, 2015] Sweewarman Balachandran and Ella M. Atkins. A constrained markov decision process for flight safety assessment and management. In *2015 AIAA Infotech @ Aerospace (5-9 January, Kissimmee, Florida)*. American Institute of Aeronautics and Astronautics, January 2015.

[Bazzan, 2009] Ana LC Bazzan. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *AAMAS*, 18(3):342–375, 2009.

[Blackmore *et al.*, 2010] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *Robotics, IEEE Transactions on*, 26(3):502–517, 2010.

[Bnaya *et al.*, 2013] Zahy Bnaya, Roni Stern, Ariel Felner, Roie Zivan, and Steven Okamoto. Multi-agent path finding for self interested agents. In *Sixth Annual Symposium on Combinatorial Search*, 2013.

[Chung, 2001] K. L. Chung. *A Course in Probability Theory*. Academic Press, 2001.

[Demir *et al.*, 2014] Nazli Demir, Behcet Acikmese, and Matthew W Harris. Convex optimization formulation of density upper bound constraints in markov chain synthesis. In *American Control Conference, 2014*, pages 483–488. IEEE, 2014.

[Demir *et al.*, 2015] Nazlı Demir, Utku Eren, and Behçet Açıkmeşe. Decentralized probabilistic density control of autonomous swarms with safety constraints. *Autonomous Robots*, 39(4):537–554, 2015.

[Dresner and Stone, 2008] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, pages 591–656, 2008.

[El Chamie *et al.*, 2016] Mahmoud El Chamie, Yue Yu, and Behcet Acikmese. Convex synthesis of randomized policies for controlled markov chains with density safety upper bound constraints. In *American Control Conference, 2016*, July 2016.

[Feyzabadi and Carpin, 2014] Seyedshams Feyzabadi and Stefano Carpin. Risk-aware path planning using hirerachical constrained markov decision processes. In *Automation Science and Engineering, 2014 IEEE International Conference on*, pages 297–303. IEEE, 2014.

[Greenshields *et al.*, 1935] BD Greenshields, Ws Channing, Hh Miller, et al. A study of traffic capacity. In *Highway research board proceedings*, volume 1935. National Research Council (USA), Highway Research Board, 1935.

[Hausknecht *et al.*, 2011] Matthew Hausknecht, Tsz-Chiu Au, and Peter Stone. Autonomous intersection management: Multi-intersection optimization. In *Intelligent Robots and Systems, 2011 IEEE/RSJ International Conference on*, pages 4581–4586. IEEE, 2011.

[Hazelton and Watling, 2004] Martin L Hazelton and David P Watling. Computation of equilibrium distributions of markov traffic-assignment models. *Transportation Science*, 38(3):331–342, 2004.

[Hsu *et al.*, 2006] Shun-Pin Hsu, Ari Arapostathis, and Ratnesh Kumar. On optimal control of Markov chains with safety constraint. In *American Control Conference (ACC) 2006*, pages 1–6. IEEE, 2006.

[Hu *et al.*, 1998] Junling Hu, Michael P Wellman, et al. Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, volume 98, pages 242–250. Citeseer, 1998.

[Nain and Ross, 1986] Phillippe Nain and Keith W Ross. Optimal priority assignment with hard constraint. *Automatic Control, IEEE Transactions on*, 31(10):883–888, 1986.

[Ono *et al.*, 2013] Masahiro Ono, Yoshiaki Kuwata, and J Balaram. Mixed-strategy chance constrained optimal control. In *American Control Conference, 2013*, pages 4666–4673. IEEE, 2013.

[Peeta and Ziliaskopoulos, 2001] Srinivas Peeta and Athanasios K Ziliaskopoulos. Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, 1(3-4):233–265, 2001.

[Puterman, 1994] Martin L Puterman. Markov decision processes: Discrete dynamic stochastic programming. *New York, NY: John Wiley*, 10:DOI: 9780470316887, 1994.

[Watling and Cantarella, 2015] David P Watling and Giulio E Cantarella. Model representation & decision-making in an ever-changing world: the role of stochastic process models of transportation systems. *Networks and Spatial Economics*, 15(3):843–882, 2015.

[Xiao *et al.*, 2014] Nan Xiao, Xuehe Wang, Lihua Xie, Tichakorn Wongpiromsarn, Emilio Frazzoli, and Daniela Rus. Road pricing design based on game theory and multiagent consensus. *Automatica Sinica, IEEE/CAA Journal of*, 1(1):31–39, 2014.