Research Note
RN/09/02

Model Checking Optimisation-Based Congestion Control Models

Alessio Lomuscio
Department of Computing
Imperial College London, UK

Ben Strulo and Nigel Walker
BT Research
Adastral Park, UK

Peng Wu
University College London
Adastral Park, UK

17 March 2009

Contents

List of Tables

List of Figures

Model Checking Optimisation-Based Congestion Control Models

Alessio Lomuscio
Department of Computing, Imperial College London, UK

Ben Strulo and Nigel Walker
BT Research Adastral Park, UK

Peng Wu
University College London, Adastral Park, UK

Abstract

Model checking has been widely applied for verification of network protocols, particularly on the sequences of interactions between protocol entities. Alternatively, optimisation has been used to reason about the large scale dynamics of networks, particularly with regard to congestion and rate control protocols such as TCP. This paper intends to provide a bridge and explore synergies between these two approaches. An optimisation-based congestion control algorithm was usually being assumed as synchronous in the sense that all agents act on the same time schedule, while some literature like [1] suggested the existence of asynchronous algorithms that have been proved stable. We carry over the duality structure of the underlying optimisation models and explore systematically these options of composition frameworks with formal semantics. The nondeterminism of interleaving between active agents represents discrete approximations of those optimisation-based congestion control algorithms with a hierarchy of expressiveness. We then use branching-time temporal logic to specify formally the convergence criteria for the system dynamics. Through model checking, convergence of those algorithms can be witnessed in the form of the sequences of interactions between primal and dual agents. Moreover, the number of steps for a system to converge at a stable equilibrium point can also be quantified. However, these discrete transition models cannot always converge due to discretisation and relaxation of synchrony, but the counterexamples returned from the model checker NuSMV [2] reveal certain realistic issue where the continuous state space is a less fit. We report on our experiences in using the abstractions of model checking to capture features of the continuous dynamics typical of optimisation-based approaches.

1 Introduction

Model checking has been widely applied to reason about network protocols in terms of the sequences of interactions between protocol entities. This typically allows the discovery of functional problems in a network protocol, such as whether the protocol can deadlock or otherwise fail to achieve the desired outcome.

Congestion control protocols generally exhibit fairly simple behaviour when considered from this perspective. But their design also faces other concerns which are typically analysed in a rather different framework. For example, TCP (Transmission Control Protocol) allows a very large number of communication sources to interact with network resources in such a way that all the sources get a fair and efficient share of the resources, especially when the resource usage of each source is averaged over multiple packet round trip times. Thus, each source sends packets at such a rate that most packets can be successfully transmitted, even though it has to compete with many other sources for the capacity of the resources along the routes it is using.

To analyse and design such a congestion control protocol, optimisation-based approaches describe the protocol in the term of a convex optimisation problem over a communication network. This is typically to maximise the value of the network allocation to its users but constrained by the available capacity of the network.

The detailed behaviour of TCP is concerned with the state of a source as packets (particularly acknowledgement packets) arrive and data packets are sent. But optimisation-based approaches abstract this to the continuous time scale over which we can measure the *rate* of the source sending packets along a route through the network, as a continuous positive real number variable. The protocol is then specified as an algorithm which defines how the rate should change as feedback (actually in the form of acknowledgements providing congestion information) reaches the source.

This algorithm then gives rise to a set of trajectories in the continuous state space. Analysis is typically concerned with the *stability* of these trajectories, that is, for any topology and as many demands as possible, whether the protocol can drive the network into close to its optimum.

Our work seeks, through two case studies, to explore how the composition and nondeterminism features of model checking can be used to represent and check features of this sort of protocols and how the alternative models can give different perspectives on the behaviours of the protocols. The main contributions are:

- *To show how the structures of the optimisation models can be naturally carried over into the transition models for model checking.*

  The sources and resources, instead of concrete protocol entities, are modelled symbolically in accordance with the duality structure of the underlying optimisation models.

- *To demonstrate a range of ways to use the composition and nondeterminism of the transition models to represent features of real systems, no matter whether the features have already been captured in the optimisation models or not.*

  A modelling spectrum is presented for composing those source and resource agents as a system, ranging from fully synchronous models to fully asynchronous models and various combinations of them. Unlike the underlying optimisation models, these transition models support uncertain gain and propagation delay, and nondeterministic congestion control schemes.

- *To illustrate ways in which the optimisation and transition models can be seen as complementary and to suggest further research to explore this complementarity.*

  We investigate the similarities, differences and synergies between the optimisation and model checking approaches by first applying model checking approaches directly on a particular congestion control protocol [3], which has already been well analysed using optimisation-based approaches. We then consider a variant of the same problem where optimisation-based approaches are weaker precisely because of the existence of significant discrete behaviour in the modelled protocol.

  The model checking results reveal the diversity on the stability of the transition models experimented, due to discretisation and relaxation of the fully synchronous structure. The counterexamples returned from the model checker NuSMV [2] illustrate a realistic reason of instability when the continuous state space is a less fit.

We started the work with NuSMV due to its full support for CTL and LTL, as well as explicit fairness constraints. We have also tried, but not reported here, other model checkers, such as SPIN [4] and UPPAAL [5], with no better performance in other slightly different settings.

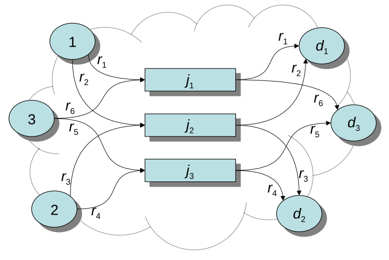
The rest of the paper is organised as follows. Section 2 briefly introduces the two congestion control models. Section 3 presents the modelling spectrum with expressiveness analysis for each composition framework. The stability property is formulated in Section 4, followed by the experimental model checking results. The paper is concluded in Section 5.

Related Work Optimisation-based approaches are now standard for analysing congestion control, starting with [6]. [3] proposed a stable fluid-flow framework for joint routing and rate control on which our first example is based. [7, 8] are similar works both using a Lagrangian optimisation-based model, which decomposes into distributed synchronous and asynchronous algorithms for congestion control.

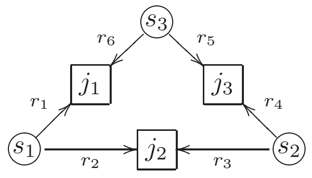
On the other hand, formal verification techniques were introduced into network research. [9] applied PROMELA/SPIN to verify the equilibrium property of a priority pricing-based congestion control model. [10] presented an extended compositional network simulation environment with the capability of bounded model checking.

Our work inherits the global viewpoint of optimisation-based approaches and characterises the stability property in a logical manner. This makes it different from literature on model checking network protocols, where concrete protocol entities such as border gateways and interior routers were modelled and their local functionalities were the main concern. Our work also differs from [9] in that we discretise and analyse a continuous optimisation-based congestion control model and explore the issue of nondeterminism in this setting, while the model checking result in [9] is just applicable for the particular model considered.

The work closest to our own is the asynchronous algorithm presented in [1], which was based on optimisation but schedules the sources and resources in a nondeterministic order. The stability of this algorithm has been proved under certain assumptions.



(a) Network Topology



(b) Resource Topology

Figure 1: A Communication Network

2 Optimisation-Based Congestion Control

This section will briefly present an optimisation formulation of congestion control, specially in a multi-path setting. We consider a network where each communication source can choose from a number of pre-configured routes to satisfy its transmission demand. This offers potential for obtaining maximal throughput and improving network resilience. In such systems, sources adjust flow rates of data on their own routes to maximise the utility derived from transmission, while avoiding or decreasing congestion at resources that have data flow rate limits.

Assume a network with a set S of sources and a set J of resources. Let R be a set of routes, each identifying the non-empty subset of resources used by the route. Let $r \in s$ denote that source s can transmit along route r and $s(r)$ be the unique source s such that $r \in s$. Let x_r be the flow rate on route $r \in R$ and C_j be the capacity of resource $j \in J$. Set $A_{jr} = 1$ if $j \in r$ and set $A_{jr} = 0$ otherwise. For instance, in a network shown in Figure 1(a), each source $i(= 1, 2, 3)$ transmits data to its destination d_i along two routes. So $S = \{s_i \equiv (i, d_i) \mid i = 1, 2, 3\}$ and $R = \{r_1, \cdots, r_6\}$. Figure 1(b) presents the resource topology of the network, in which each source owns two routes (i.e., $r_{2i-1} \in s_i$ and $r_{2i} \in s_i$ for $i = 1, 2, 3$), and each resource is shared by two routes (i.e., $j_1 \in r_1, j_1 \in r_6$ and $j_i \in r_{2(i-1)}, j_i \in r_{2i-1}$ for $i = 2, 3$).

Let $\mathtt{x}$, C and A be the corresponding vectors and incidence matrix, that is, $\mathtt{x} = (x_r, r \in R)$, $C = (C_j, j \in J)$ and $A = (A_{jr}, j \in J, r \in R)$. Let x_j denote the aggregate flow rate at resource j. A resource j is *congested* if $x_j > C_j$ (recall that C_j is the capacity of resource j). A route r is *congested* if j is congested for some $j \in r$.

Then, the multi-path congestion control problem is typically specified as the following optimisation problem:

$$\max \sum_{s \in S} U_s(\sum_{r \in s} x_r) \text{ such that } A\mathtt{x} \leq C, \mathtt{x} \geq 0 \tag{1}$$

where U_s is a utility function of source s. Assume that for each $s \in S$, U_s is increasing and strictly concave in its argument. The problem can then be solved by finding a saddle point of the following Lagrangian function:

$$L(\mathtt{x}, \mathtt{y}) = \sum_{s \in S} U_s(\sum_{r \in s} x_r) - \mathtt{y}(A\mathtt{x} - C) \tag{2}$$

where $\mathtt{y} = (y_j, j \in J)$, $\mathtt{y} \geq 0$ and $\mathtt{x} \geq 0$. This is because if $(\mathtt{x}^*, \mathtt{y}^*)$ is a saddle point of L, then $\mathtt{x}^*$ is an optimal solution to (1) [1].

The $\mathtt{x}$ and $\mathtt{y}$ in the Lagrangian are referred to as *primal* and *dual* decision variables, respectively. Each primal (dual) variable selects its value on behalf of a source (resource) such as to maximise (minimise) the value of $L(\mathtt{x}, \mathtt{y})$, given the values of all other variables. Thus, a solution to problem (1-2) is where the interactions between the sources and resources reach an equilibrium [7, 8].

An optimisation-based approach would then specify an algorithm modelled as a set of trajectories in the primal and dual decision variables. The analysis would typically demonstrate provable convergence of these trajectories to the optimum under certain modelling and applicability assumptions [6, 1, 3, 7, 8, 11].

The rest of this section will present two congestion control models based on the above multi-path setting. Both models are meant to balance resource allocation amongst the competing sources to achieve an efficient network utilisation, while not starving any particular sources.

For the first model (Multi-path congestion/rate control), we intend to capture the fluid-flow congestion control algorithm specified in [3] by Kelly et.al. Under the fluid flow assumption the algorithm was described by the trajectories of differential equations and supported by proof of stability. We apply a direct discretisation to this model and choose particular functions to make it tractable for model checking. This discretisation exhibits variant behaviour that is apparently unrealistic for the system that Kelly et.al. models. But the process of discretisation leads us to consider the meaning of nondeterminism present in discrete (transition) models, which does not exist in those differential equations.

Our second model (Session-based rerouting and termination), is inspired by this experience and considers a scenario where the assumptions behind the Kelly et.al.'s model start to fail and where the assumptions of model checking become more realistic. We choose a system where flow rates are discrete-valued, corresponding to the discrete nature of model checking, and the behaviour of a source is itself naturally nondeterministic.

2.1 Multi-path congestion/rate control

In [3] a protocol was presented that gives rise to trajectories in the primal flow rates x_r as a continuous function of time t, i.e., $x_r(t)$, subject to the following differential equation:

$$\frac{d}{dt} x_r(t) = \kappa_r x_r(t) \left(1 - \frac{y_r(t)}{U'_{s(r)}(x_{s(r)}(t))} \right)^{+}_{x_r(t)} \tag{3}$$

where κ_r is a constant and

- $y_r(t) = \sum_{j \in r} y_j(t)$ is the total cost on route r; $y_j(t)$ is the cost at resource j;

- $x_s(t) = \sum\limits_{r \in s} x_r(t)$ is the aggregate flow rate on all routes serving source s;
- U_s is a utility function of source s and U_s' is its first-order derivative.
- $(z)_x^+ = \min(0, z)$ if $x \leq 0$, $(z)_x^+ = z$ otherwise.

These dynamic equations actually solve not the exact optimisation problem (1) but an approximation in which there is an explicit cost y_j for each resource j as its load approaches its capacity. As long as this cost function is chosen appropriately it has only a small effect on the equilibrium values of the system but significantly improves its controllability. It also can be interpreted as a real cost (for example from packet delay) that arises from operating network resources too close to their capacities.

In [3] propagation delay in a network was taken into account by defining y_r and x_s as functions of the past route flow rates. Herein, we omit this consideration and assume propagation delay to be negligible. We will come back to this point in Section 5.

Now we simply discretise the Kelly et.al.'s model by making the time discrete and making state variables integer-valued under integer arithmetic, and by choosing particularly tractable instances of the generic functions in Equation (3).

When the continuous time parameter t is abstracted into a discrete one, the flow rate function $x_r(t)$ has to be converted into a series of instantaneous snapshots of x_r. This also applies to $y_r(t)$ and $x_s(t)$. The relation between the current value of x_r and its next value x_r' can be defined uniformly as $x_r' = x_r + \Delta x_r$, where Δx_r is the increment of x_r in one unit time.

We choose y_j to be a linear function of the flow rates at resource j, that is,

$$y_j = \beta_j x_j \quad \text{with} \quad x_j = \sum_{j \in r} x_r \tag{4}$$

where β_j is a price coefficient. Following a rather common choice, we assume U_s to be a logarithmic function of the aggregate flow rate on all routes serving source s, that is,

$$U_s = \alpha_s \ln(x_s) \quad \text{with} \quad x_s = \sum_{r \in s} x_r \tag{5}$$

where α_s is a utility coefficient.

Then, by following the skeleton of Equation (3) with (4) and (5), Δx_r is defined as

$$\Delta x_r = \kappa_r x_r \left(1 - \frac{1}{\alpha_s} \sum_{j \in r} \beta_j x_j \sum_{r' \in s(r)} x_{r'} \right)_{x_r}^{+} \tag{6}$$

Here, κ_r can be regarded as a *gain* coefficient that defines the pace at which route r seeks its equilibrium; while $\{\frac{\beta_j}{\alpha_s}\}_{j \in r}$ defines the saddle point where route r will settle. Especially, the lower bound of x_r for each route r is set to 1 unless it is initialised to be zero. A more accurate constraint was applied in [3]. This is to avoid a *spurious* convergence trace, where the aggregate flow rates at all resources reach zero.

Equations (6) and (4) define how the sources and resources act, respectively. In order to define the complete behaviour of an algorithm, we also have to define how these actions are composed (or scheduled). The agents in the Kelly et.al.'s algorithm can be thought of as acting synchronously though in infinitesimal steps. In a discrete model composition structures that do not constrain those actions to synchrony could also be applicable, as suggested by [1]. We will explore these options in Section 3.

Once we allow this relaxation, then the sources no longer proceed with the deterministic gain implied by κ_r. Although κ_r is constant, those asynchronous models allow the sources to update their flow rates in a nondeterministic order. This then may lead the routes to equilibrate at a variable pace, which is significantly different to the deterministic behaviour specified by Equation (3).

2.2 Session-based rerouting and termination

In the second model, we consider essentially the same underlying optimisation problem but with the context moved to a regime where a continuous real-valued model is a less reliable fit. We consider a system where the sources are managing a non-empty set of constant flow rate sessions. Two control actions are provided for a source to resolve its possible congestion status.

Firstly, the source may reroute sessions to an alternative route. Rather than considering a deterministic rerouting policy, we will let the model explore how the source may choose new routes for the excess sessions on congested routes.

Secondly, the sources may, though only *in extremis*, terminate those excess sessions. This follows the general ideas specified in the IETF Pre-Congestion Notification (PCN) Working Group [12], where an architecture for controlling congestion through admission control and flow termination is being defined. For each route, its source relies on feedback from its destination to determine the excess flow rate to be terminated. The destination will inform the source the proportion of congestion at the bottleneck resource (if any). The source then uses this feedback to decide how many proportion of its flow to terminate.

For the second model, the primal variable x_r of problem (1) can be regarded as the number of sessions on a route, a more naturally discrete value. We take a linear utility function U_s, that is,

$$U_s = \alpha_s x_s \quad \text{with} \quad x_s = \sum_{r \in s} x_r \tag{7}$$

where α_s is the utility value of a single session of source s. This choice sounds appropriate for a network operator who typically treats all sessions equally.

Then the congestion control policy of the second model is as follows: for each congested route $r \in s$, source s will divert a certain number of excess sessions from route r to a non-congested route $r' \in s$; or terminate them if such r' does not exist.

The proportion of sessions to be rerouted or terminated is based on the proportions of excess load at resources, that is, for resource j,

$$y_j = \frac{x_j - C_j}{x_j} \quad \text{with} \quad x_j = \sum_{j \in r} x_r \tag{8}$$

Then, for a congested route $r \in s$,

$$\Delta x_r = -x_r \max\{y_j \mid j \in r\} \tag{9}$$

Herein, $\max\{y_j \mid j \in r\}$ is the largest proportion of congestion at certain bottleneck resource j.

For a non-congested route $r' \in s$, $\Delta x_{r'}$ is the aggregate flow diverted to r' from those congested routes $r \in s$.

As before these equations define how the primal and dual decision variables will change. The possible composition structures will be explored later. However, this model is nondeterministic even in the synchronous case. This is because whenever there is more than one non-congested route available, source s will choose one of them nondeterministically for each congested route $r \in s$.

Remark 1. Although we have not tried yet, hybrid model checking seems indeed applicable in the present setting. We believe it would emulate the standard optimisation-based approaches to a much closer extent (at a computational cost). However, it seems likely that it would confirm, among other things, the relationship between nondeterminism and stability discussed in the paper.

3 Modelling

Based on optimisation-based congestion control models, distributed algorithms were developed, depending on whether it is the sources (primals), the resources (duals), or both controlling the evolution of a system actively, as well as the types of interleaving of their control actions. Most of these algorithms scheduling the sources and/or resources synchronously along the unique continuous time scale [1, 3, 8], while an asynchronous algorithm was also presented in [1], which schedules the sources and resources in a nondeterministic order. In the rest of this section we will explore the options of these variant composition structures through the means of formal semantics.

Note that it is the data flows in a network that are concerned by those optimisation models. So the behaviours of the sources and resources, which control and monitor the data flows, respectively, are prescribed in corresponding congestion control algorithms, but not the detailed hop-by-hop behaviours of particular protocol entities. Thus, we follow the perspective of optimisation-based approaches, that is, to encode the sources and resources as procedural agents.

Technically we adopt a special form of symbolic transition graph with assignments (STGA) [13], termed as *symbolic assignment graph* (or SAG as in the following), to model the systems above. The explicit input/output constructs in STGA is omitted, due to the fact that shared variables can be relied on for this purpose. Recall that symbolic transition graphs are a basic symbolic semantics for value-passing CCS, π-calculus, and others. Herein, the notion of SAG has the benefit of offering a succinct semantics to describe the systems above, which is amenable to model checking; clearly other formalisms are also possible. We recall the basic constructions below but we refer to [13] for more details.

We presuppose the following syntactic categories:

- *Val* is a set of values, ranged over by v;
- *Var* is a set of variables, ranged over by x;
- *Exp* is a set of data expressions over $Val \cup Var$, ranged over by e;

- *BExp* is a set of boolean expressions ranged over by b;

An *assignment* θ has the form $\bar{x} := \bar{e}$, where $\bar{x}$ (respectively $\bar{e}$) represents a list of variables (respectively data expressions), with $\bar{x}$ and $\bar{e}$ having the same length. Assume $Assign_V$ be the set of all assignments to variables in $V \subseteq Var$.

A *valuation* ρ is a total mapping from *Var* to *Val*. Applications of ρ onto data expression e and boolean expression b are denoted by $\rho(e)$ and $\rho(b)$ as usual. Especially, we write $\rho \vDash b$ if $\rho(b) = true$. The valuation $\rho\{\bar{x} \mapsto \bar{v}\}$ is same as ρ except mapping $\bar{x}$ to $\bar{v}$. Let $\theta\rho$ denote the resulting valuation by applying assignment θ onto ρ, that is, $\theta\rho \equiv \rho\{\bar{x} \mapsto \rho(\bar{e})\}$. Assume *Eval* be the set of all valuations.

Definition 3.1 (Symbolic Assignment Graph)**.** Given a set of variables $V \subseteq Var$, a *symbolic assignment graph* (SAG) is a tuple $M_V = (Q, \mathcal{T}, q^0)$ where

- Q is a set of symbolic states;
- $\mathcal{T} \subseteq Q \times BExp \times Assign_V \times Q$ is a set of symbolic transitions, each $(q, b, \bar{x} := \bar{e}, q') \in \mathcal{T}$ denoted by $q \xrightarrow{b,\bar{x}:=\bar{e}} q'$ with $\{\bar{x}\} \subseteq V$;
- $q^0 \in Q$ is the initial symbolic state.

Informally a symbolic transition $q \xrightarrow{b,\bar{x}:=\bar{e}} q'$ denotes a possible state change of the SAG from q to q', under the assumption that the guard b is evaluated to true at state q, and in doing so the values of $\bar{x}$ are changed to the ones of $\bar{e}$ evaluated at state q. The following definition makes this precise.

Definition 3.2 (Labelled Transition Systems)**.** Given a set of observable labels $L_V = \{\mu_W \mid W \subseteq V\}$, and an initial valuation ρ_0, the concrete semantics of a SAG M_V is a labelled transition system $[\![M_V]\!]_{\rho_0} = (P, T, p^0)$, where

- $P = \{q_\rho \mid q \in Q, \rho \in Eval\}$ is a set of states;
- $T \subseteq P \times L_V \times P$ is the least set of transitions given by the following operational rule:

$$\frac{q \xrightarrow{b,\bar{x}:=\bar{e}}, q'}{q_\rho \xrightarrow{\mu_{\{\bar{x}\}}} q'_{\theta\rho}} \quad \rho \vDash b$$

  Similarly we write $p \xrightarrow{\mu_W} p'$ for each $(p, \mu_W, p') \in T$ with $W \subseteq V$;
- $p^0 \equiv q^0_{\rho_0} \in P$ is the initial state.

To conclude we say that $\omega = \mu_{X_1} \cdots \mu_{X_i} \cdots \in L_V^*$ is a *trace* of M_V if there exists a sequence of transitions $q^0_{\rho_0} \xrightarrow{\mu_{X_1}} q^1_{\rho_1} \cdots q^{i-1}_{\rho_{i-1}} \xrightarrow{\mu_{X_i}} q^i_{\rho_i} \cdots$ in $[\![M_V]\!]_{\rho_0}$ for some initial valuation ρ_0.

In what follows we explore source and resource agents modelled by symbolic assignment graphs (or SAGs as in the following). To define the evolution of a system we use the standard notions of synchronous and asynchronous compositions.

Definition 3.3 (Synchronous Composition)**.** Given two symbolic assignment graphs $M_{V_1} = (Q_1, T_1, q_1^0)$ and $M_{V_2} = (Q_2, T_2, q_2^0)$ with $V_1 \cap V_2 = \emptyset$, $M_{V_1}|M_{V_2}$ is the graph $M_{V_1 \cup V_2} = (Q_1 \times Q_2, T, (q_1^0, q_2^0))$, where T is the least set of transitions given by the following operational rule:

$$\frac{q_1 \xrightarrow{b_1, \bar{x}_1 := \bar{e}_1} q_1' \qquad q_2 \xrightarrow{b_2, \bar{x}_2 := \bar{e}_2} q_2'}{(q_1, q_2) \xrightarrow{b_1 \wedge b_2, (\bar{x}_1, \bar{x}_2 := \bar{e}_1, \bar{e}_2)} (q_1', q_2')}$$

Definition 3.4 (Asynchronous Composition)**.** Given two symbolic assignment graphs $M_{V_1} = (Q_1, T_1, q_1^0)$ and $M_{V_2} = (Q_2, T_2, q_2^0)$ with $V_1 \cap V_2 = \emptyset$, $M_{V_1} \parallel M_{V_2}$ is the graph $M_{V_1 \cup V_2} = (Q_1 \times Q_2, T, (q_1^0, q_2^0))$, where T is the least set of transitions given by the following operational rules:

$$\frac{q_1 \xrightarrow{b_1, \bar{x}_1 := \bar{e}_1} q_1'}{(q_1, q_2) \xrightarrow{b_1, \bar{x}_1 := \bar{e}_1} (q_1', q_2)} \qquad\qquad \frac{q_2 \xrightarrow{b_2, \bar{x}_2 := \bar{e}_2} q_2'}{(q_1, q_2) \xrightarrow{b_2, \bar{x}_2 := \bar{e}_2} (q_1, q_2')}$$

The state defined in our optimisation models is precisely the values of the primal and dual variables: the flow rates $\{x_r \mid r \in R\}$ and congestion costs $\{y_j \mid j \in J\}$. The agents who control this state are sources and resources respectively. We take this structure over directly to our SAG-based transition models which contain source agents and resource agents.

In scheduling these agents, three forms of optimisation-based congestion control algorithms exists. One standard form is termed primal algorithms, in which the sources actively adjust their primal variables and the resources simply immediately recalculate the values of their dual variables. Another one is termed dual algorithms, in which it is the resources that take the active parts. Finally the other is termed primal/dual algorithms, in which both classes of agents are active. In the rest of the section we model the cases of primal algorithms and primal/dual algorithms; the case of dual algorithms can also be modelled but we do not present it here for lack of space.

3.1 Congestion control as SAGs - primal algorithms

In the case of primal algorithms the sources are active agents modelled as SAGs, while the resources are modelled by deterministic functions recalculating the congestion costs passively. Thus we are assuming that the resources respond with the up-to-date congestion information quickly on the timescale at which the primal agents are taking their actions. This does not allow a resource agent to apply more complex algorithms to smooth the congestion information it sends: it has to be a simple function of the current load at the resource.

For each source $s \in S$, let $X_s = \{x_r \mid r \in s\}$ denote the source agent's decision variables. In this framework, we associate a SAG M_{X_s} updating its own variables X_s at each transition. We assume that a source updates all of its own variables simultaneously in one transition, which is counted as one source update. For the multi-path congestion/rate control model described in Section 2.1, this leads us to the following SAG: for a source s owning $k \geq 1$ routes $r_1, \ldots, r_k$:

$$M_{X_s} = (\{q\}, \{q \xrightarrow{true,\ x_{r_1}, \cdots, x_{r_k} := x_{r_1} + \Delta x_{r_1}, \cdots, x_{r_k} + \Delta x_{r_k}} q\}, q)$$

where $\Delta x_{r_i}(1 \leq i \leq k)$ is defined by equation (6).

Having defined source updates we are left with two options to represent the interleaving of all source agents in this framework: by synchronously composition generating a system of Synchronous Sources (or SS), or by asynchronous composition generating a system of Asynchronous Sources (or AS):

$$SS \triangleq \Big|_{s \in S} M_{X_s}$$
$$AS \triangleq \Big\|_{s \in S} M_{X_s}$$

These two compositions model different scenarios. Intuitively, SS inherits the synchronous structure of dynamic equations like (3). But as a discrete model, it reflects the case where propagation delay periods are uniform on every route.

AS constitutes the general case of naturally modelling the concurrent nature of a distributed network, where the sources are monitored in uncertain paces.

3.2 Congestion control as SAGs - primal/dual algorithms

In this subsection we model both sources and resources agents as SAGs and consider their compositions. Modelling resources agents explicitly fully reflects the delayed reaction of the resources to source updates. Recall that resources' decision variables are the dual variables $\{y_j \mid j \in J\}$ in the Lagrangian optimisation problem (2).

Similarly to the previous section for each resource $j \in J$, we associate a resource agent M_{y_j} modelled as a SAG built on y_j as its state variables. Resource agents may be composed synchronously (respectively asynchronously), thereby generating systems of Synchronous (respectively Asynchronous) resources or SR (AS respectively).

$$SR \triangleq \Big|_{j \in J} M_{y_j}$$
$$AR \triangleq \Big\|_{j \in J} M_{y_j}$$

Combining this analysis with the one of the previous subsection, we obtain four general modelling frameworks, in which a particular class of sources agents are asynchronously composed with a particular class of resources agents.

$$SSSR \triangleq SS \parallel SR$$
$$SSAR \triangleq SS \parallel AR$$
$$ASSR \triangleq AS \parallel SR$$
$$ASAR \triangleq AS \parallel AR$$

The modelling options above will in general produce different global evolutions of the systems modelled. However, there is certain redundancy in the modelling above. For instance, consecutive source updates on disjoint subsets of decision variables could usefully be treated jointly as a synchronous source update as they are all based on the same (current) states of resource agents and one source update will not affect others. Alternatively these source updates may be treated asynchronously in any particular interleaving order and always lead to the same state as done by executing the (equivalent) synchronous update.

This also applies to consecutive resource updates on disjoint subsets of decision variables. This obviously opens the possibility of employing state reduction techniques, notably partial order reduction, when checking a whole system.

3.3 Expressiveness

In this subsection, the expressiveness of each modelling framework will be illustrated through trace analysis. It will be seen that different interleaving structures of these modelling frameworks associate optimisation models with distinct senses of nondeterminism.

With the above observations, we extend the notation μ_W to represent trace fragments. Let $X = \bigcup\limits_{s \in S} X_s$ and $Y = \{y_j \mid j \in J\}$. For some $S' = \{s_1, \cdots, s_l\} \subseteq S, l \geq 1$, let $W = \bigcup\limits_{s \in S'} X_s$ and μ_W represent the synchronous updates on $\{x_r \mid r \in s, s \in S'\}$, or any permutation of the sequence of asynchronous updates $\mu_{X_{s_1}} \cdots \mu_{X_{s_l}}$, whichever applicable. Similarly, for some $W' = \{y_{j_1}, \cdots, y_{j_m}\} \subseteq Y, m \geq 1$, let $\mu_{W'}$ represent the synchronous updates on $\{y_j \mid j \in W\}$, or any permutation of the sequence of asynchronous updates $\mu_{\{y_{j_1}\}} \cdots \mu_{\{y_{j_k}\}}$. Let $\mathcal{C}_X$ and $\mathcal{C}_Y$ be the set of compound source and resource update labels, respectively, i.e., $\mathcal{C}_X = \{\mu_W \mid W \subseteq X$ and for each $s \in S$, $W \cap X_s = \emptyset$ or $W \cap X_s = X_s\}$ and $\mathcal{C}_Y = \{\mu'_W \mid W' \subseteq Y\}$.

Then, the traces of these modelling frameworks can be expressed as regular expressions shown in Table 2(a).

Models	Traces
SS	$(\mu_X)^*$
AS	$(\{\mu_{X_s} \mid s \in S\})^*$
$SSSR$	$(\mu_X \mid \mu_Y)^*$
$SSAR$	$(\mu_X \mid \mathcal{C}_Y)^*$
$ASSR$	$(\mathcal{C}_X \mid \mu_Y)^*$
$ASAR$	$(\mathcal{C}_X \mid \mathcal{C}_Y)^*$
AS^*	$(\mathcal{C}_X \mu_Y)^*$

(a)

$$\begin{tikzcd}
 & ASAR \\
SSAR \arrow[ur, "\prec"'] & ASSR \arrow[u, "\prec"'] \\
SSSR \arrow[u, "\prec"'] \arrow[ur, "\prec"'] & AS^* \arrow[u, "\prec"'] \\
SS \arrow[u, "\prec"'] \arrow[ur, "\prec"'] & AS \arrow[u, "\prec"']
\end{tikzcd}$$

(b)

Figure 2: Modelling Spectrum

Let $traces(M)$ be the set of traces of agent M. Then, $M \preceq N$ if $traces(M) \subseteq traces(N)$, and $M \prec N$ if $traces(M) \subset traces(N)$. Thus, the hierarchy shown in Figure 2(b) can be derived immediately from Table 2(a). Note that these trace inclusion relations may be naturally inferable from the semantics of synchronous and asynchronous composition. But they actually clarify in detail the difference of these modelling frameworks in representing the interactions between the sources and resources.

1. $SSSR \prec SSAR \prec ASAR$ and $SSSR \prec ASSR \prec ASAR$.

   Asynchronous source agents can nondeterministically choose to ignore resource updates, while synchronous ones cannot. Thus changing to asynchronous agents makes the rate at which the sources move towards equilibrium more uncertain.

By including the resources explicitly as agents, these models can partially capture an uncertain propagation delay between the sources and resources, in the sense that one source (or resource) update will not take effect until the connected resources (or sources) act and thereby pass on the information.

2. $SS \prec SSSR$ and $AS \prec ASSR$.

   Since only the sources are active in SS and AS models, only source update labels appear in their traces. But if we incorporate the state change of resources' decision variable explicitly, the *full* traces for SS and AS models can be written as $(\mu_X \mu_Y)^*$ and $(\{\mu_{X_s} \mid s \in S\}\mu_Y)^*$, respectively, where μ_Y represents evaluating the resource functions. Thus, SS and AS models can be regarded as having *fast* resources, which immediately react to all source updates. So this is equivalent to a primal/dual model in which there is a synchronous resource update after every source action. The difference between SS and AS then is that between the resource updates, in SS all sources act, while in AS only one source does.

3. $AS^* \prec ASSR$

   AS^* is an asynchronous composition of asynchronous source agents and synchronous resource agents, restricted with an alternative scheduling policy between sources and resources. That is, source and resource updates are arranged in a turn-based mode. In each turn, only sources (or resources) get updated, followed by resources (or sources) getting updated in the next turn.

   For AS^* (like SS and AS), each source update will take effect on all resources before the next one. That is, the resources will not miss any source updates. On the contrary, $ASSR$ allows consecutive parallel source updates, which can be interpreted as allowing the sources to update faster than the resources do.

As can be seen from the above trace patterns, these modelling frameworks can capture the effect of network propagation to certain extent. But since all agents always share the same view on the global state, these models cannot capture situations in which propagation delay leads the sources to act together but on an inconsistent view of the states of resources.

Remark 2. We were interested in whether nondeterminism from asynchrony would allow us to make statements about the behaviours of the congestion control protocols that are independent of propagation delay encountered by signalling mechanisms.

However, it seems that, because the state space of a system is simply the product of the state spaces of its component agents, we cannot model the *on-the-fly* message states between agents. These additional states make the transition models considerably more complex but are also required if we need to model the situation when different agents can take inconsistent snapshots of the states of other agents.

A natural way to represent continuous propagation delay would be to augment the input/output (I/O) constructs of STGA with extra data facilities, such as queues. We do not pursue this here. But we observe that in the absence of fully specified queueing behaviour, the I/O constructs on their own do not help to model propagation delay since the synchronous semantics of the I/O constructs (which implements rendezvous communication) excludes any delay; while the asynchronous semantics does not preserve the correct propagation order when a sequence of outputs occurs.

4 Verification

Following the optimisation-based congestion control policies presented in Section 2, the transition model frameworks presented in Section 3 suggest a series of synchronous or asynchronous (gradient projection) algorithms that might be applicable to solve the multi-path congestion control problem in a distributed network. The machinery of an optimisation approach will then establish the effectiveness of such an algorithm by proving the trajectories in the primal and dual decision variables converge to the optimum objective value. The transition models of these algorithms open the possibility of applying model checking to verify their convergence properties. Indeed, note that it is straightforward to associate a NuSMV process to any SAG described above, and that synchronous and asynchronous compositions may equally be implemented through standard machinery in NuSMV. In this section we report on the lessons learnt from a series of experiments we have run in this setting.

4.1 Specifications

Recall from Section 2 that the optimisation problem (1) can be solved by analysing the convergence property of the Lagrangian function $L(\mathtt{x}, \mathtt{y})$ of equation (2). A saddle point of $L(\mathtt{x}, \mathtt{y})$ is a solution to the problem. We are interested in properties expressing whether the sources (and the system as a whole) have reached optimality, whether the flows have stabilised, and, if so, whether the capacity constraints are satisfied.

With the presence of nondeterminism in an optimisation-based congestion control algorithm (such as uncertain gain and propagation delay, random rerouting policy, etc.), model checking is a natural choice for verification. Moreover, following a perturbation from an equilibrium (perhaps due to a fault), we might be interested to know not only whether the algorithm will reconfigure the network flow to a new optimum, but also how quickly it does so. Although the objective function of problem (1) itself does not consider the convergence time, this can also be investigated through model checking.

Herein, we express these properties as particular specifications in CTL [14]. Note that while in general convergence is naturally expressed as a SAT modulo theory problem [15], for the case being discussed here we consider convergence of the objective function to a given value u^*, i.e., $\sum_{s \in S} U_s(\sum_{r \in s} x_r) = u^*$. Given the fact that the domain of each x_r is bounded, we can then infer the range of the objective function. So, to check whether there exists one or more values u^* to which the objective function may converge, we can repeatedly run model checking experiments exploring the range of u^* with the binary search strategy. If the objective function converges then the CTL formula $AG \sum_{s \in S} U_s(\sum_{r \in s} x_r) =$

u^* holds at some future state for some value u^* in the range. Note that this is a stability rather than an optimality condition.

In our experiments we checked the following specifications:

$$AF\ AG \quad \sum_{s\in S} U_s(\sum_{r\in s} x_r) = u^* \tag{10}$$

$$EF\ AG \quad \sum_{s\in S} U_s(\sum_{r\in s} x_r) = u^* \tag{11}$$

Specification (10) states that the objective function of problem (1) always converges to some value u^*, while Specification (11) states that it does so along at least one trace.

The Lagrangian function (2) also concerns convergence of each route flow rate, which would lead to a saddle point of the function. So we are also interested in considering the following specifications:

$$AF\ AG \quad ((\sum_{s\in S} U_s(\sum_{r\in s} x_r) = u^*) \wedge \bigwedge_{r\in R} (x'_r = x_r)) \tag{12}$$

$$EF\ AG \quad ((\sum_{s\in S} U_s(\sum_{r\in s} x_r) = u^*) \wedge \bigwedge_{r\in R} (x'_r = x_r)) \tag{13}$$

Recall that x'_r is the next value of x_r.

4.2 Experimental results

The combination of the individual agent transitions described in Section 2 with the composition rules in Section 3 gives a collection of models that can be straightforwardly coded into NuSMV, the model checker we used in our experiments, and which we ran on a Xeon Dual-Core 64-bit 2.8GHz machine with 1GB memory.

To do this we need to fix the topology of the network considered. We chose to investigate symmetric resource topologies where each source controls multiple routes, every route comprises only one shared resource, and every resource is shared by multiple sources. The results we present in this paper, summarised in Table 1, are for the network shown in Figure 1, which has three sources and three resources. The capacity of each resource was set to 6.

The column *#Reachable_states* shows the numbers of the reachable states of these transition models. Due to the data-oriented nature of the underlying optimisation models, it can be seen that the reachable state space grows dramatically, though as expected, from synchronous models to asynchronous models.

For the twelve models defined in Table 1, Specifications (10) and (12) were found not satisfiable, that is, each of the models was found to be unstable. However, most models show the network does converge in some traces, that is, there exist values for the constant u^* resulting in satisfiable Specifications (11) and/or (13).

The penultimate column u^**(Stable)* reports the validity of Specification (13), and hence Specification (11) for each model; while for this case the column *#Min_steps* shows the minimal number of control actions that each model takes to reach an equilibrium.

The column u^**(Vibrating)* shows the values of the constant u^* for which Specification (11) but not Specification (13) is satisfied.

(a) Multi-path Congestion/Rate Control

No.	Composition	Congestion	#Reachable_states	u^*(Stable)	#Min_steps	u^*(Vibrating)
1	SS	Route Overload	23	NONE		NONE
2	SS	Resource Failure	17	NONE		NONE
3	AS^*	Route Overload	82723	?		?
4	AS^*	Resource Failure	6187	ln(100)	?	NONE
5	$ASSR$	Route Overload	2.06719e+06	?		?
6	$ASSR$	Resource Failure	35445	ln(100)	8	NONE

(b) Session-based Rerouting and Termination

No.	Composition	Congestion	#Reachable_states	u^*(Stable)	#Min_steps	u^*(Vibrating)
7	SS	Route Overload	3	NONE		NONE
8	SS	Resource Failure	7	NONE		14
9	AS^*	Route Overload	16	16	2	18
10	AS^*	Resource Failure	1418	11-12	3	13,15,16
11	$ASSR$	Route Overload	8513	*12-18*	2	*13-18*
12	$ASSR$	Resource Failure	32200	*4-12*	5	13-18

Table 1: Model Checking Results

The convergence results of these transition models are analysed in detail below.

4.2.1 Multi-path congestion/rate control

Our first batch of experiments were based on the congestion control example with primal transition rule (6) and dual update rule (4), with $\alpha_s = 36\beta_j$, $k_r = 2$ for each s, j, r. Two different initial conditions were used: *Route Overload* and *Resource Failure*. The first, with $\mathtt{x} = (5\ 3\ 3\ 3\ 1\ 3)$, was chosen to emulate a situation in which the network must redistribute load because one route has excess load which can be carried elsewhere. In the second we chose $\mathtt{x} = (0\ 3\ 3\ 3\ 3\ 0)$ with the additional constraint that resource j_1 cannot carry traffic to emulate a resource failure. The model checking results are in Table 1(a).

Figure 3(a) and 3(b) illustrate the traces that witness convergence of Model 4 (AS^*) and 6 ($ASSR$), respectively, with respect to Specification (13). In what follows, the utility coefficient in the objective function (5) are not relevant. So we check convergence of $\sum\limits_{s \in S} \ln x_s$ to some constant value, that is, $EF\ AG\ \sum\limits_{s \in S} \ln x_s = u^*$. Through the experiments we find that Specification (10) is not satisfiable for any u^*.

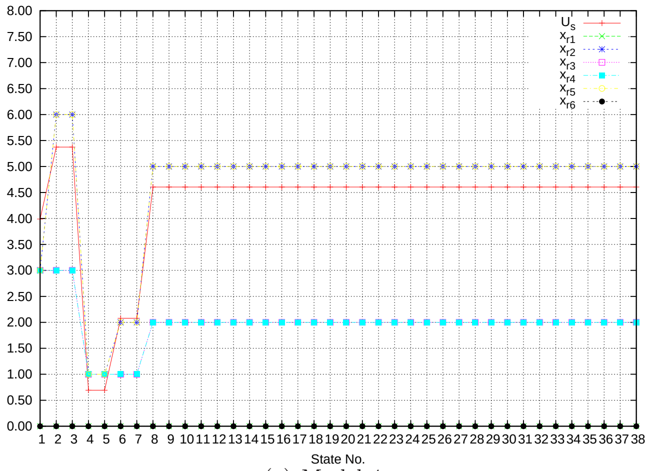
Because the individual transition rules are deterministic, their synchronous composition (SS) leads to a deterministic trace, and relatively few states. However, the discretisation of the continuous state space leads to a limit cycle rather than a final stable state, and model checking has picked this up in the failure to satisfy Specification (10) and Specification (11). Figure 4 shows the instability cases that happen in Model 1 and 2, respectively. This also applies to the AS^* and $ASSR$ models.

The instability of these models is actually caused by too much gain in each primal update. Given a larger domain of route flow rates, the gain coefficient κ_r for each route r in Equation (3) can be decreased. This results in a finer discretisation. Figure 5 illustrates convergence of Model 1 and 2, respectively, with smaller gain coefficients. The capacity of each resource was set to 60 and $\kappa_r = k$ for each route r. This also applies to the AS^* and $ASSR$ models. With finer discretisations, Specification (10) and (12), or their weaker forms, are satisfiable to these models for a single value or a set of values of the constant u^*. The stability results of SS and $AS^*/ASSR$ models conform to Theorem 1 and 2 in [1], respectively. Please refer to Appendix A for details.

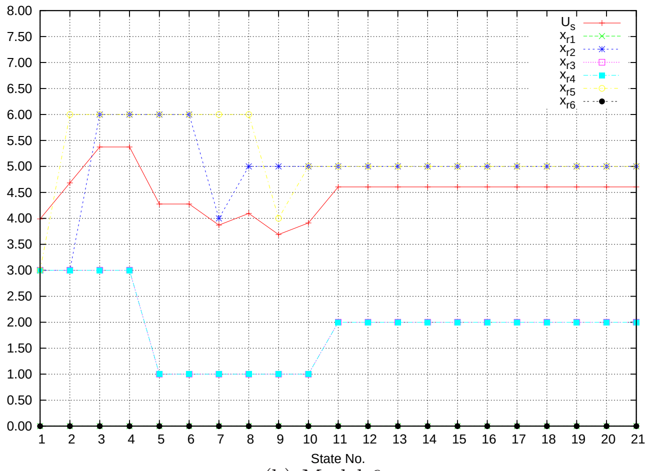
The above convergence traces show that the local imbalances can be corrected in a few number of steps, either by the local sources (the ones affected by the faults) themselves, or by spreading them to affect further away sources. Figure 5 suggests that with a finer discretisation, this could happen more smoothly with reducing amplitudes, as happened in [3], rather than in such a jagged way that all sources converge *suddenly*, as shown in Figure 3.

Convergence in this setting mirrors the results of the original optimisation models in terms of differential equations. However, by means of model checking we can here quantify quite precisely the amount of control actions, hence the time, required for the network to recover from a perturbation.

As the interleaving constraints are relaxed the number of accessible states increases, suggesting a severe instability in the system. This is not unexpected since it is now possible for one source agent to act many times before the others



(a) Model 4

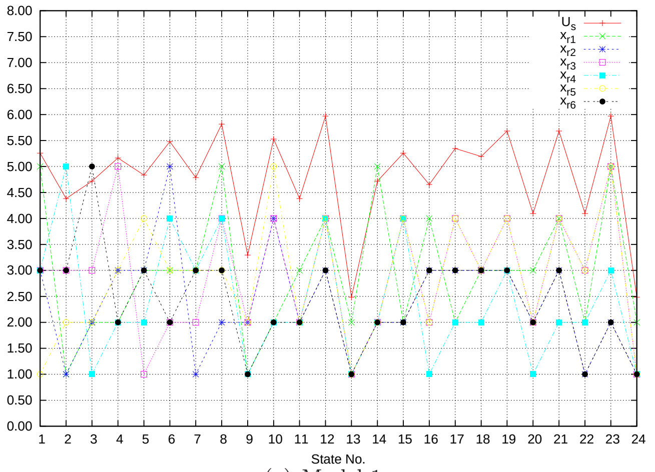


(b) Model 6

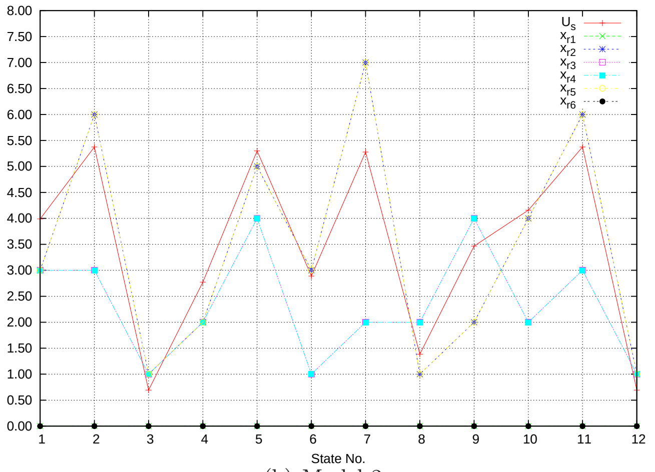
Figure 3: Multi-path Congestion/Rate Control - AS^* & $ASSR$ Convergence

do: repeated actions on route r, when projected back into the optimisation framework, corresponds roughly to an increase in the gain coefficient k_r, and increasing gain within a feedback loop typically leads to instability.

A composition of the agents which faithfully captures the optimisation dynamics of Equation (3) but which also allows certain nondeterminism in the interleaving of the agents is therefore not well represented in the options we have investigated in Table 1. It appears that some notion of fairness is required that is intermediate between (i) complete synchrony and (ii) forcing each agent always to act eventually. Developing a well motivated correspondence between optimisation inspired dynamics and model checking requires both a notion of interleaving fairness and the quantisation of the state space to be taken into account together, as they can both be seen as related to the gain parameter in the optimisation models. We did not pursue this question further in our experiments.



(a) Model 1

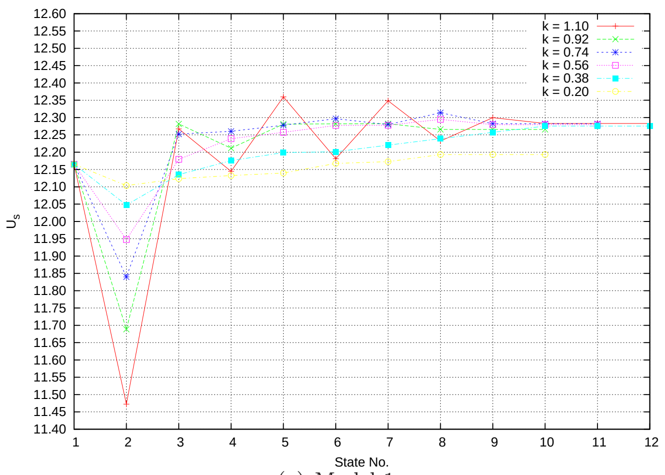


(b) Model 2

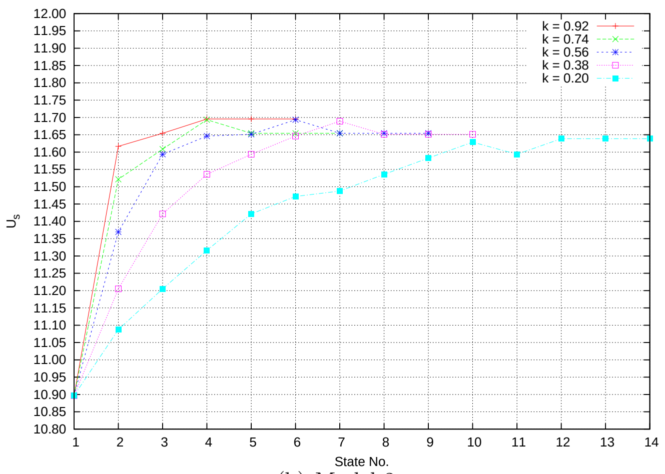
Figure 4: Multi-path Congestion/Rate Control - Instability

4.2.2 Session-based rerouting and termination

In the light of the above observations we chose our second batch of experiments to be based on a scenario that is closer to the natural idiom of model checking. Here transition rules are not designed to correspond to smooth optimisation dynamics in any way. Instead they are designed to terminate flows in ways comparable to real systems like [12]. We use the dual update rule (8) and the primal update rule (9), where each resource j has a maximum capacity C_j equal to 6. The primal rule features nondeterminism in which route it chooses to reallocate flow to. Again, we consider two types of initial conditions: *Route Overload* and *Resource Failure*. The first corresponds to starting in a state with one route overloaded by 2 units, such that $\mathtt{x} = (5\ 3\ 3\ 3\ 1\ 3)$, with duals calculated from (8). The second initial condition is designed to emulate the situation immediately after a resource failure, so we set $\mathtt{x} = (3\ 3\ 3\ 3\ 3\ 3)$, but set $C_{j_1} = 0$. The model checking results are in Table 1(b).



(a) Model 1

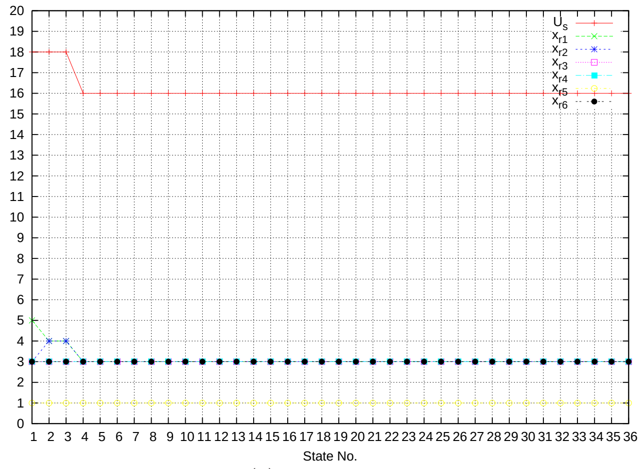


(b) Model 2

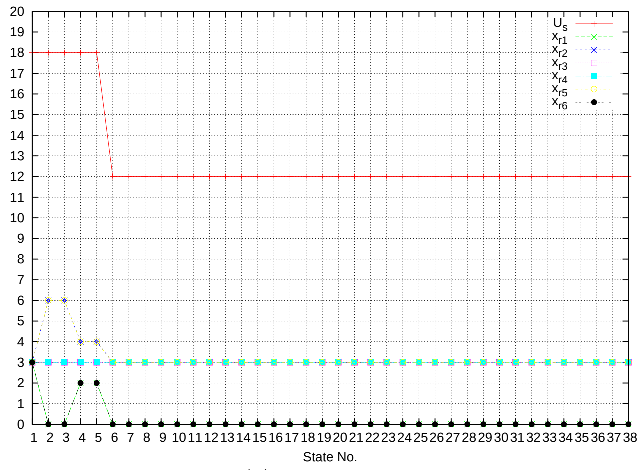
Figure 5: Multi-path Congestion/Rate Control - SS Convergence

The traces illustrated in Figure 6 and Figure 7 identify the reported convergence of the utility function and all the route flow rate variables $\{x_r \mid r \in s\}$. Because the transition rules are designed only to shed load rather than to increase it we do not see any state explosion corresponding to instability in any of the composition scenarios. Similarly we consider herein convergence of $\sum_{s \in S} x_s$, that is, $EF\ AG\ \sum_{s \in S} x_s = u^*$.

The last column ("u^*(Vibrating)") reports traces that satisfy Specification (11) but not Specification (13), as shown in Figure 8(b). In these traces the utility function converges to the value shown but at least some of the $x_r, r \in s$ do not meet the capacity constraint and continue to change ("vibrate"). In these circumstances, the system is entering a limit cycle in which the total load offered into the network is greater than the capacity that it can carry, but no individual source knows it must terminate some flow. Instead they pass the excess flow



(a) Model 9



(b) model 10

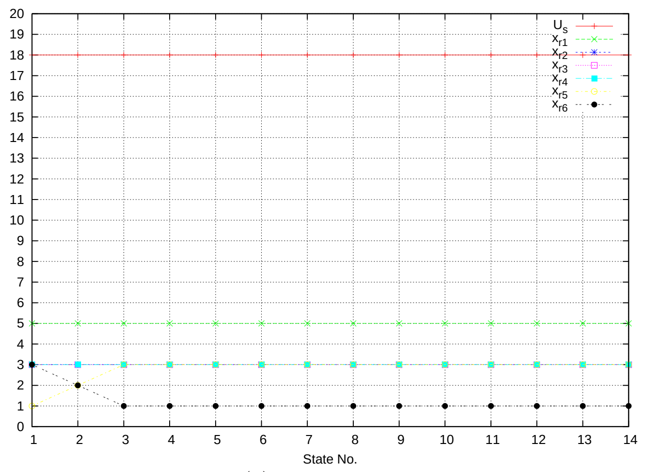
Figure 6: Session-based Rerouting and Termination - AS^* Convergence

around in a cycle. A trace showing this is presented in Fig 8. This behaviour seems possible in real systems and therefore model checking has detected a real possible issue with this simple design. A possible approach to resolve this is to ensure that a real system has enough asynchrony so that the flow termination algorithm will behave properly[1].

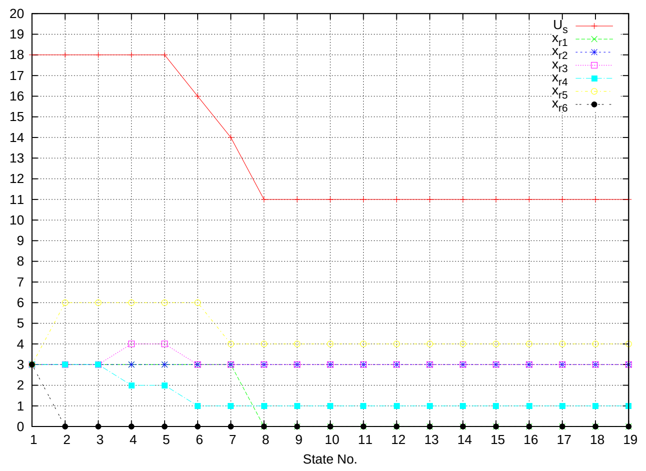
The AS^* and $ASSR$ models of the Session-based Rerouting and Termination system abstractly implement the single-marking proposal of [16], but run under a multi-path setting. Thus, they both reflect the impacts of multi-path routing (where traffic may be diverted due to resource failures) and flash crowds (where a large burst of admission request may be accepted at once and then overload the resources).

Remark 3. In these two case studies, the $ASSR$ models may terminate more

[1]A technical solution might be for the sources to look for rerouting loops and initiate some random terminations once they are detected.



(a) Model 11



(b) model 12

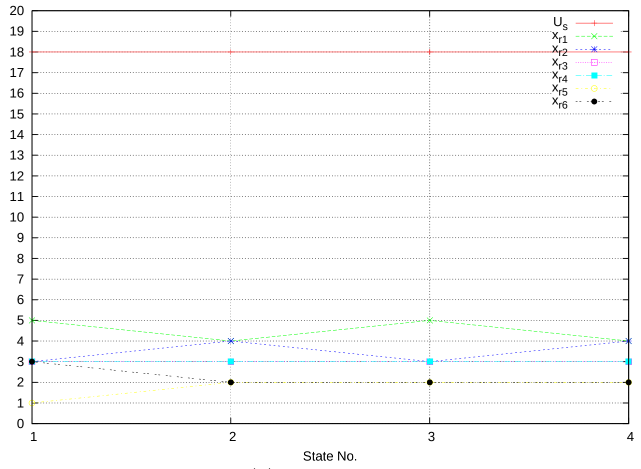
Figure 7: Session-based Rerouting and Termination - $ASSR$ Convergence

flows than the AS^* models do to converge. This suggests a performance degradation when the sources update faster than resource and therefore miss a few resource updates.

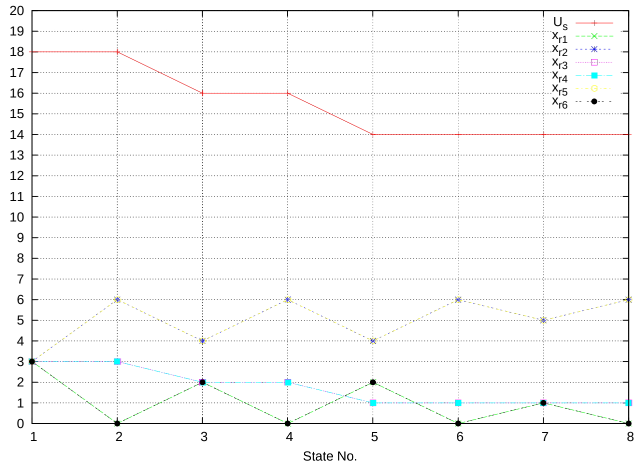
Remark 4. Compared to the models of the Multi-path Congestion/Rate Control, the models of the Session-based Rerouting and Termination system can lead to convergence by terminating more flows. The results also show that there exists a non-trivial lower bound on the number of excess flows to be terminated.

5 Discussion

In translating from optimisation based continuous dynamics to model checking we identified two possibilities for the interpretation of nondeterminism. In the first it represents the choice that each agent has *within* each action it takes.



(a) Model 7



(b) Model 8

Figure 8: Session-based Rerouting and Termination - Instability

From an optimisation point of view this type of choice arises when the solution to a local problem is not unique: if two routes have equal least cost then the total flow can be split or moved arbitrarily between them. In the second interpretation nondeterminism represents choice of *sequencing* of the actions of the agents which can be derived from the implementation of the agents, and our first scenario suggested that allowing too much nondeterminism in the agent implementation would lead to system instabilities.

Another way of thinking about nondeterminism is that it accounts for loss of knowledge in moving from real systems to abstract models. We had hoped that this abstraction would allow us to make statements about the behaviour of congestion control schemes that are independent of the propagation delay encountered by signalling mechanisms (indeed the resource matrix used in our model is already forgetful of details of the underlying resource connectivity). However, this was not straightforward. The difficulties arise from modelling the additional state actually present in the propagating messages. We did build some models, not reported on here, in which we explicitly represented state 'on-the-fly' within the signalling system (or within input queues and buffers). Our hope was that the increase in the system state space could be offset by reducing the explosion of possible evolutions due to interleaving semantics. In other words we produced larger but more deterministic models. However, our initial experiments still ran into size limitations of the model checkers that we used (NuSMV and UPPAAL).

In the standard modelling idiom, the state space is the product of the state space of all the individual agents (sources and resources in our case). Crucially this idiom abstracts away from the real state information on-the-fly. This same abstraction is also implicit in the optimisation based smooth dynamics of (3). In that case the agents are assumed to act sufficiently *slowly* for the abstraction to be valid. In model checking, by contrast, the agents are assumed to act instantaneously, but sufficiently *infrequently* for it to be valid. In both cases this assumption could be interpreted either as a limitation on the accuracy of the model (if analysing a system), or as constraints on implementation, or as conditions that must be policed by some other mechanism in the network (if synthesising a system). In both cases the assumption is quite brittle. In optimisation dynamics it has been shown that an arbitrarily small delay can render an otherwise stable system unstable [17]. In the model checking idiom an arbitrarily small delay could allow a sequence of transitions not captured by the delay free semantics. While the optimisation and discrete models appear at first sight to be quite different, in their modelling of delay it turns out that they share very similar types of limitations.

6 Conclusions

The paper presents a way to integrate optimisation-based approaches with model checking techniques. On one hand, it associates optimisation models with nondeterminism; on the other hand, it associates the structure of optimisation-based approaches into model checking. A spectrum of modelling frameworks are presented importing different granularities of nondeterminism: uncertain gain and propagation delay, and nondeterministic rerouting.

We believe that logic methods and model checking should offer machinery that complements optimisation theory in the design and analysis of network control processes We have investigated this proposition in the context of dynamic allocation of traffic amongst multiple routes across a network, a topic that is attracting some attention within the networking research community. Our experiments showed some promise in this direction, but have highlighted some limitations. Not surprisingly we were limited to small concrete topologies by the state explosion problem. We see one way of addressing this could be to combine theorem proving and model checking techniques. However, they also highlighted more subtle points concerning the interpretation and specification of the interleaving semantics.

1. the transition models lack accuracy of error, due to the limit of data types supported by the model checker.

2. the notion of continuous propagation delay relies on a history of global states, which is not represented in our modelling paradigms.

3. the state space explosion problem still exists, though partial order reduction technique can help eliminate irrelevant interleaving in the transition models.

4. Unlike the optimisation models, the transition models are topology-dependant.

As part of future work, we would like further investigate techniques to improve the above weaknesses, including hybrid model checking (that can handle real-valued variables and continuous functions), modelling paradigms with queues (that can represent continuous propagation delay) and other model checking techniques (assume/guarantee based compositional model checking, bounded model checking, directed model checking, etc.)

Also it would be interesting to evaluate the performance aspects of these optimisation models in the nondeterministic context introduced by our modelling paradigms, such as the probability or the time for a faulty network to converge.

References

[1] Low, S.H., Lapsley, D.E.: Optimization flow control, I: basic algorithm and convergence. IEEE/ACM Transactions on Networking (TON) **7**(6) (1999) 861–874

[2] Cimatti, A., Clarke, E.M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella., A.: NuSMV 2: An opensource tool for symbolic model checking. In: Proceedings of the 14th International Conference on Computer-Aided Verification (CAV'02), Copenhagen, Denmark (2002)

[3] Kelly, F., Voice, T.: Stability of end-to-end algorithms for joint routing and rate control. ACM SIGCOMM Computer Communication Review **35**(2) (2005) 5–12

[4] Holzmann, G.J.: The model checker SPIN. IEEE Transactions on Software Engineering **23**(5) (1997) 279–295

[5] Bengtsson, J., Larsen, K.G., Larsson, F., Pettersson, P., Yi, W.: Uppaal — a Tool Suite for Automatic Verification of Real–Time Systems. In: Proceedings of Workshop on Verification and Control of Hybrid Systems III. Number 1066 in Lecture Notes in Computer Science, Springer–Verlag (1995) 232–243

[6] Kelly, F., Maulloo, A., Tan, D.: Rate control for communication networks: shadow prices, proportional fairness and stability. Journal of the Operational Research Society **49** (1 March 1998) 237–252(16)

[7] Walker, N., Wennink, M.: Interactions in transport networks. Electronic Notes in Theoretical Computer Science **141**(5) (2005) 97–114

[8] Wennink, M., Williams, P., Walker, N., Strulo, B.: Optimisation-based overload control. In: NET-COOP. (2007) 158–167

[9] Yuen, C., Tjioe, W.: Modeling and verifying a price model for congestion control in computer networks using promela/spin. In: Proceedings of the 8th International SPIN Workshop on Model Checking of Software (SPIN'01), New York, NY, USA, Springer-Verlag New York, Inc. (2001) 272–287

[10] Sobeih, A., Viswanathan, M., Hou, J.C.: Check and simulate: a case for incorporating model checking in network simulation. In: Proceedings of the 2nd ACM and IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE'04). (2004) 27–36

[11] Strulo, B., Walker, N., Wennink, M.: Lyapunov convergence for lagrangian models of network control. In: NET-COOP. (2007) 168–177

[12] Eardley, P.: Pre-congestion notification (PCN) architecture. Internet-Draft draft-ietf-pcn-architecture-08 (2008)

[13] Lin, H.: Symbolic transition graph with assignment. In: Proceedings of the 7th International Conference on Concurrency Theory (CONCUR'96), LNCS 1119, Springer-Verlag (1996) 50–65

[14] Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems (TOPLAS) **8**(2) (1986) 244–263

[15] Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving sat and sat modulo theories: From an abstract davis-putnam-logemann-loveland procedure to DPLL(T). Journal of the ACM **53**(6) (2006) 937–977

[16] Charny, A., Zhang, X., , Faucheur, F.L., Liatsos, V.: Pre-congestion notification using single marking for admission and termination. Internet-Draft (2007)

[17] Voice, T.: Stability of multi-path dual congestion control algorithms. In: Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools (VALUETOOLS '06), New York, NY, USA, ACM (2006) 56

A Finer Discretisation

[1] presented two optimisation-based flow control algorithms, one is synchronous and the other is asynchronous. The synchronous algorithm is close to our SS models except that it regards the resource updates as state transitions, not functions. The asynchronous algorithm is close to our $ASSR$ models except that it allows the sources and resources to update at the same time.

Both algorithms in [1] were proved to be stable and to converge to the optimum under the following assumptions:

1. On given intervals, the utility functions U_s are increasing, strictly concave, twice continuously differentiable; the curvatures of U_s are bounded away from zero;

2. The step size is small enough;
3. For all sources and resources, the time between consecutive updates is bounded.

Actually the third assumption was only applied in the proof of the optimality of the asynchronous algorithm. Bases on these results, we experiment the SS, AS^* and $ASSR$ models of the Multi-path Congestion/Rate Control system with a finer discretisation setting, that is, the capacity C_j of each resource j is 24, which makes the route flow rates x_r ranged in $[0, 25]$, while the gain coefficient κ_r is set to 0.2 for each route r. This experiments are performed on a cluster with two Xeon 4-Core 64-bit 2.5GHz CPUs and 16GB memory. The model checking results are in Table 2, where Model k' is the revision of Model k under the finer discretisation setting. As before, the two initial scenarios are concerned: *Route Overload* with $\mathtt{x} = (20\ 12\ 12\ 12\ 4\ 12)$ and *Resource Failure* with $\mathtt{x} = (0\ 12\ 12\ 12\ 12\ 0)$.

The column $\{u^*\}$*(Stable)* shows the set U^* of constant values such that each model satisfies the following specification:

$$AF\ (\bigvee_{u^* \in U^*} AG\ (\sum_{s \in S} U_s(\sum_{r \in s} x_r) = u^*)) \tag{14}$$

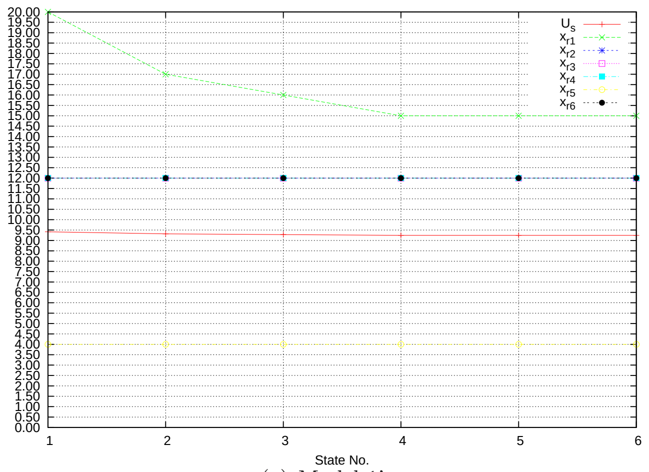
The above specification is a weaker form of Specification (10). It means that the objective function of problem (1) always converges to some value u^*, but such u^* may not be unique.

It can be seen that for the SS and AS^* models, Specification (14) reduces to Specification (10). As shown in Figures 9 and 10, the AS^* models converge to the same equilibrium point as the corresponding SS models do.

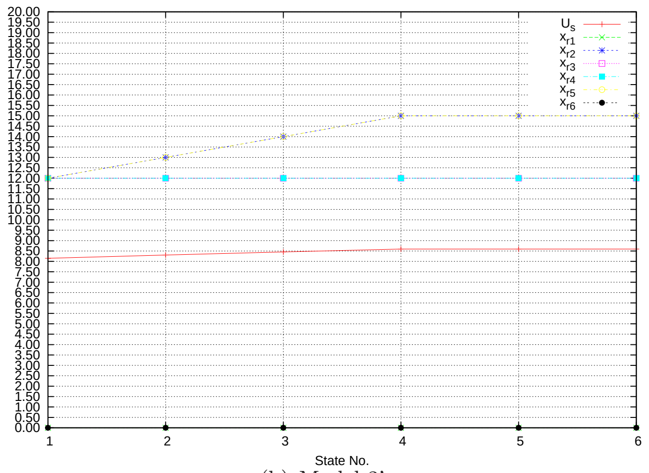
However, this does happen to the $ASSR$ models. Note that the $ASSR$ models are constrained with a fairness condition that every source and resource agent is activated infinitely often. This fairness condition implies the above third assumption because no specific boundary was defined. But this does not guarantee the uniqueness of the equilibrium point, as shown in Figure 11.

No.	Composition	Congestion	#Reachable_states	$\{u^*\}$(Stable)	#Min_steps	u^*(Vibrating)
1'	SS	Route Overload	4	ln(10368)	3	NONE
2'	SS	Resource Failure	4	ln(5400)	3	NONE
3'	AS^*	Route Overload	11	ln(10368)	3	NONE
4'	AS^*	Resource Failure	114	ln(5400)	3	NONE
5'	$ASSR$	Route Overload	18	ln(9216)	5	NONE
				ln(9600)	4	
				ln(9984)	3	
				ln(10368)	2	
6'	$ASSR$	Resource Failure	225	ln(5400)	6	NONE
				ln(5760)	7	
				ln(6120)	8	
				ln(6144)	8	
				ln(6528)	9	
				ln(6936)	10	

Table 2: Multi-path Congestion/Rate Control - Finer Discretisation ($k_r = 0.2$)

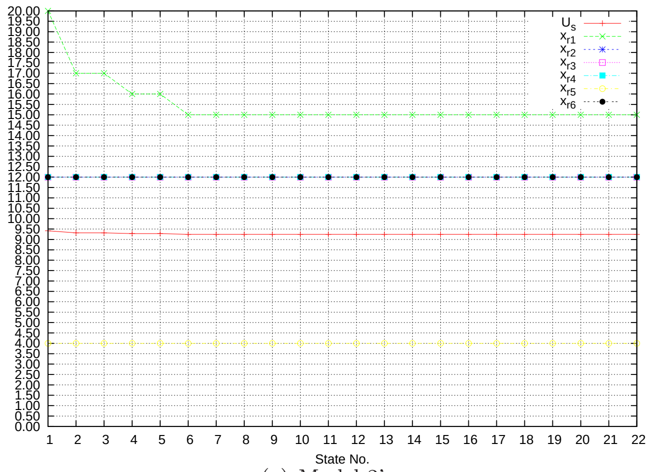


(a) Model 1'

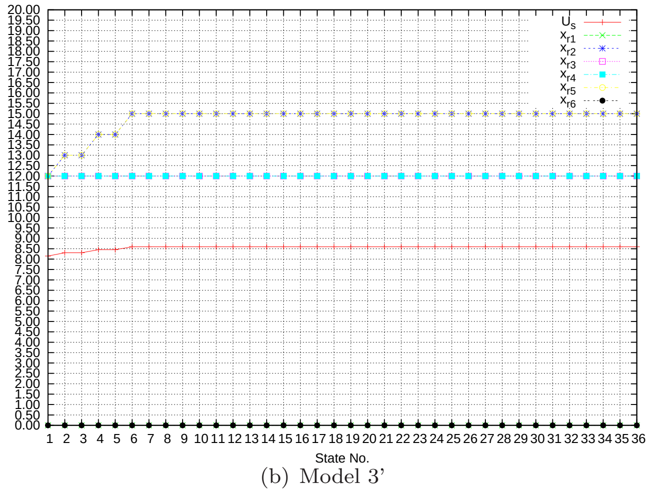


(b) Model 2'

Figure 9: *SS* Convergence

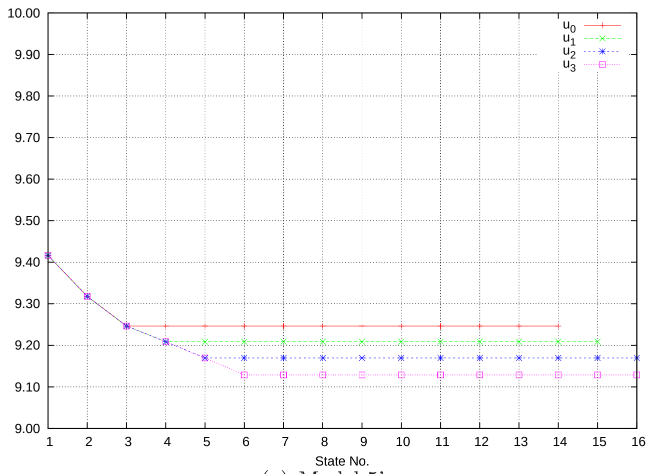


(a) Model 2'

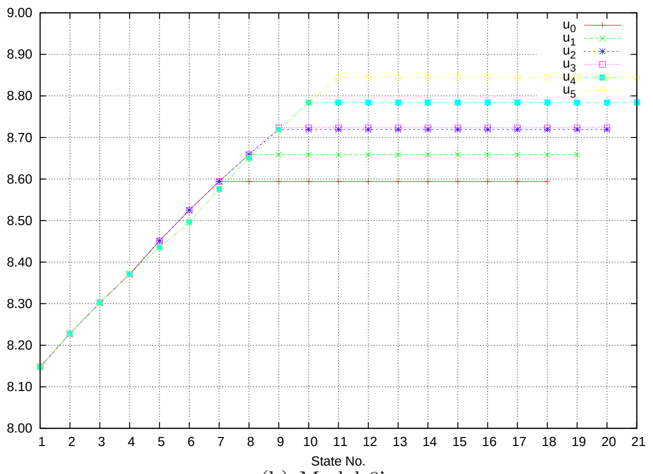


(b) Model 3'

Figure 10: AS^* Convergence



(a) Model 5'



(b) Model 6'

Figure 11: *ASSR* Convergence