Model Checking Coalition Nash Equilibria in MAD Distributed Systems

Federico Mari[1], Igor Melatti[1] ⋆, Ivano Salvo[1], Enrico Tronci[1], Lorenzo Alvisi[2], Allen Clement[2], and Harry Li[2]

[1] Dep. of Computer Science, University of Rome "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
{mari,melatti,salvo,tronci}@di.uniroma1.it
[2] Dep. of Computer Science, University of Texas at Austin
1 University Station C0500, Austin, Texas, USA
{lorenzo,aclement,harry}@cs.utexas.edu

Abstract. We present two OBDD based model checking algorithms for the verification of Nash equilibria in finite state mechanisms modeling *Multiple Administrative Domains* (MAD) distributed systems with possibly colluding agents (*coalitions*) and with possibly faulty or malicious nodes (Byzantine agents). Given a finite state mechanism, a *proposed protocol* for each agent and the *maximum sizes* f for Byzantine agents and q for agents collusions, our model checkers return PASS if the proposed protocol is an ε-f-q-Nash equilibrium, i.e. no coalition of size up to q may have an interest greater than ε in deviating from the proposed protocol when up to f Byzantine agents are present, FAIL otherwise. We implemented our model checking algorithms within the NuSMV model checker: the first one *explicitly* checks equilibria for each coalition, while the second represents *symbolically* all coalitions. We present experimental results showing their effectiveness for moderate size mechanisms. For example, we can verify coalition Nash equilibria for mechanisms which corresponding normal form games would have more than 5×10^{21} entries. Moreover, we compare the two approaches, and the explicit algorithm turns out to outperform the symbolic one. To the best of our knowledge, no model checking algorithm for verification of Nash equilibria of mechanisms with coalitions has been previously published.

1 Introduction

Cooperative services are increasingly popular distributed systems in which nodes (agents) belong to *Multiple Administrative Domains* (MAD). Thus in a MAD distributed system each node owns its resources and there is no central authority owning all system nodes. Examples of MAD distributed systems include Internet routing [13, 23], wireless mesh routing [18], file distribution [8], archival storage [19], cooperative backup [2, 9, 17].

⋆ Corresponding Author: Igor Melatti. Tel: +39 06 4991 8431 Fax: +39 8541842

In traditional distributed systems, nodes may deviate from their specifications (*Byzantine nodes*) because of bugs, hardware failures, faulty configurations, or even malicious attacks. In MAD systems, nodes may also deviate because their administrators are *rational*, i.e. selfishly intent on maximizing their own benefits from participating in the system (*selfish nodes*). For example, selfish nodes may change arbitrarily their protocol if that is at their advantage. *Byzantine-Altruistic-Rational* (BAR) protocols provide a realistic model for MAD systems.

Showing that a protocol P for a MAD distributed system satisfies a given specification φ entails two tasks. First, we need to show that P satisfies the given property when all rational nodes follow the protocol *exactly*. Second, we need to show that all rational nodes do, in fact, follow the protocol *exactly*.

As for the first task, well known model checking techniques (e.g. see [6] for a survey) are available to verify that a system satisfies a given property despite the presence of a limited number of Byzantine nodes. It suffices, as usual, to model Byzantine nodes with nondeterministic automata.

As for the second task, this is usually accomplished by proving that no rational agent has an incentive in deviating from the proposed protocol. This is done by proving that the proposed protocol is a *Nash equilibrium* (e.g. see [13, 4]).

A symbolic model checking algorithm to automatically verify that a given protocol is a Nash eaquilibrium for a given MAD distributed system has been presented in [20]. However the model checker presented in [20] only addresses the case in which agents do not collude. On the other hand, it is well known from game theory that coalitions of agents may have an advantage in deviating even when no single agent may get any advantage by deviating alone (e.g. see [15]). For example, this is the case for the gossip protocol presented in [16] which is a Nash equilibrium when agents do not collude (no coalitions) and instead is no longer a Nash equilibrium when *large enough* coalitions are allowed.

The above state of affairs motivates the goal of this paper: designing a model checking algorithm to verify if a given protocol is a Nash equilibrium for a MAD distributed system when coalitions up to a given size are allowed.

Our contribution. In Sect. 2 we show how a MAD distributed system with coalitions of players can be modeled as a *Coalition Schema*, that is a suitable synchronous product of *Finite State Machines*. This framework extends *Finite State Mechanisms* presented in [20] which do not account for coalitions.

In Sect. 3 we define the game induced by a Coalition Schema, and in Sect. 4 we give a formal definition of the property we want to verify: ε-f-q-Nash. Intuitively, a mechanism is ε-f-q-Nash if no coalition of size up to q of rational agents has an interest greater than $\varepsilon > 0$ (along the lines of, e.g. [12, 14]) in deviating from the *proposed protocol* when there are at most f Byzantine agents (along the lines of [11]). Sufficient conditions to verify ε-f-q-Nash property are given in Theor. 1.

In Sect. 5 we present a verification algorithm that given a coalition schema $\mathcal{Q}$, our desired precision $\delta > 0$ and (ε, f, q) as above, returns: PASS if the given mechanism is indeed a ($\varepsilon + \delta$)-f-q-Nash equilibrium for $\mathcal{Q}$, FAIL otherwise.

From a mathematical point of view, given a coalition schema $\mathcal{Q}$ and a coalition Q, we can build a mechanism $\mathcal{M}_Q$ in which the coalition is just one of the

agents. As a consequence, a first approach (that we call *explicit*) to verify that a mechanism is ε-f-q-Nash consists of adapting the symbolic algorithm in [20] to check that $\mathcal{M}_Q$ is a Nash equilibrium for all Q, such that $0 < |Q| \leq q$. If $\mathcal{Q}$ has n players, following this approach entails calling $\mathcal{O}(n^q)$ times a variation of the algorithm in [20] (more specifically, $\sum_{k=1}^{q} \binom{n}{k}$).

To overcome this exponential growth, we propose an alternative approach (that we call *symbolic*) by extending the algorithm in [20] so as to represent symbolically (i.e. using OBDDs [3]) all mechanisms $\mathcal{M}_Q$, where Q is a coalition of size at most q. We implemented our algorithm on top of NuSMV [22] using ADDs (*Arithmetic Decision Diagrams*) [10] to manipulate real valued rewards.

Finally, in Sect. 6 we present experimental results showing effectiveness of our verification algorithm on moderate size mechanisms. For example (Tab. 1 in Sect. 6), within 30 hours using 5GB of RAM we can verify Nash equilibria for mechanisms with 16 agents and coalitions of size up to 2 (i.e. 136 possible coalitions). This corresponds to find Nash equilibria for 136 games each with a normal form of more than 5×10^{21} entries.

Moreover, we compare explicit and symbolic approach performances. To this aim, we note two facts. First, our symbolic approach involves the introduction of auxiliary variables to properly perform the maximin computations required by our algorithm. Second, real valued nodes in ADDs make usual OBDD subtree sharing less effective. As a result, even if the symbolic approach should be asymptotically better, the explicit implementation outperforms the symbolic one in mechanisms we deal with, both in running time and memory usage (Tabs. 1 and 2 in Sect. 6).

Related works. Design of mechanisms for rational agents has been widely studied (e.g. [23, 21, 5]) as well as the impact of collusions (e.g. [15]). Design methods for BAR protocols have been investigated in [1, 16, 7, 11] .

We differ from such works since our focus here is on automatic verification of Nash equilibria for finite state BAR systems rather than on designing principles for them. The paper closer to ours is [20] where a symbolic algorithm for checking Nash equilibria in mechanisms has been presented. We note however that [20] does not address coalitions.

Summing up, to the best of our knowledge, no model checking algorithm for the automatic verification of Nash equilibria of finite state mechanisms with coalitions has been previously proposed.

2 Coalitions Model

In this section, we present our framework to model protocols. Finite state protocols are modeled via *Finite State Mechanisms*, which suitably extend the usual definition of the synchronous parallel composition of finite state transition systems. The notion of *Coalition Schema* (Sect. 2.2) extends the definition of Mechanism in [20] by specifying the reward function and the discount factor for each possible coalition (i.e. for each subset of players). Indeed, a Coalition Schema represents a class of Mechanisms (Sect. 2.3).

2.1 Basic Notions

We denote with $\mathbb{B}$ the set $\{0,1\}$ of boolean values (0 for *false* and 1 for *true*). We denote with $[n]$ the set $\{1,\dots,n\}$. The set of subsets of X (with cardinality at most k) will be denoted by $\mathcal{P}(X)$ ($\mathcal{P}_k(X)$).

We denote an n-tuple of objects (of any kind) in boldface, e.g. $\boldsymbol{x}$. Unless otherwise stated we denote with x_i the i-th element of the n-tuple $\boldsymbol{x}$, $\boldsymbol{x_{-i}}$ the $(n-1)$-tuple $\langle x_1,\dots,x_{i-1},x_{i+1},\dots,x_n\rangle$, and with $\langle \boldsymbol{x_{-i}}, b\rangle$ the n-tuple $\langle x_1,\dots,x_{i-1},b,x_{i+1},\dots,x_n\rangle$. Given a set $Q\subseteq[n]$, we denote the tuple $\langle \boldsymbol{x}_j\rangle_{j\in Q}$ with $\boldsymbol{x}_Q$ and the tuple $\langle \boldsymbol{x}_j\rangle_{j\notin Q}$ with $\boldsymbol{x}_{-Q}$.

2.2 Coalition Schema

Definition 1 (Coalition Schema). *An n players (agents)* coalition schema $\mathcal{Q}$ *is a tuple $\langle \boldsymbol{S},\boldsymbol{I},\boldsymbol{A},\boldsymbol{B},\boldsymbol{T},h,\beta\rangle$ which elements are defined as follows.*

$\boldsymbol{S}=\langle S_1,\dots,S_n\rangle$ *is an n-tuple of nonempty finite sets (of* local states*). The* state space *of $\mathcal{M}$ is the set (of* global states*) $S=\prod_{i=1}^{n} S_i$.*

$\boldsymbol{I}=\langle I_1,\dots,I_n\rangle$ *is an n-tuple of nonempty sets (of* local initial states*) s.t. $I_i\subseteq S_i$. The set of* global initial states *is $I=\prod_{i=1}^{n} I_i$.*

$\boldsymbol{A}=\langle A_1,\dots,A_n\rangle$ *is an n-tuple of nonempty finite sets (of* local actions*). The set of* global actions *(i.e. n-tuples of local actions) is $A=\prod_{i=1}^{n} A_i$. The set of i-*opponents actions *is $A_{-i}=\prod_{j=1,j\neq i}^{n} A_j$.*

$\boldsymbol{B}=\langle B_1,\dots,B_n\rangle$ *is an n-tuple of functions s.t., for each $i\in[n]$, $B_i : S\times A_i\times S_i\to\mathbb{B}$. Function B_i models the* transition relation *of agent i, i.e. $B_i(\boldsymbol{s},a,s')$ is true iff agent i can move from (global) state $\boldsymbol{s}$ to (local) state s' via action a. We require B_i to be* serial *(i.e. $\forall \boldsymbol{s}\in S\ \exists a\in A_i\ \exists s'\in S_i$ s.t. $B_i(\boldsymbol{s},a,s')$ holds) and* deterministic *(i.e. $B_i(\boldsymbol{s},a,s')\ \wedge\ B_i(\boldsymbol{s},a,s'')$ implies $s'=s''$). We write $B_i(\boldsymbol{s},a)$ for $\exists s'\ B_i(\boldsymbol{s},a,s')$. That is, $B_i(\boldsymbol{s},a)$ holds iff action a is allowed in state $\boldsymbol{s}$ for agent i.*

$\boldsymbol{T}=\langle T_1,\dots,T_n\rangle$ *is an n-tuple of functions s.t., for each $i\in[n]$, $T_i : S\times A_i\to\mathbb{B}$. We require T_i to satisfy the following properties: 1) $T_i(\boldsymbol{s},a)$ implies $B_i(\boldsymbol{s},a)$; 2) (*nonblocking*) for each state $\boldsymbol{s}\in\boldsymbol{S}$ there exists an action $a\in A_i$ s.t. $T_i(\boldsymbol{s},a)$ holds.*

$h:\mathcal{P}([n])\times\boldsymbol{S}\times\boldsymbol{A}\to\mathbb{R}$ *is a function that for each set $Q\subseteq[n]$, for each state $\boldsymbol{s}$ and action $\boldsymbol{a}$, gives the reward $h(Q,\boldsymbol{s},\boldsymbol{a})$ for the* coalition *Q.*

$\beta:\mathcal{P}([n])\to\mathbb{R}$ *is a function returning for coalition $Q\subseteq[n]$ a value $\beta(Q)\in(0,1)$. We call $\beta(Q)$ the discount factor of coalition Q.*

The transition relation B_i models the *underlying* behavior for agent i, that is all possible behaviors of a Byzantine agent or possible choices of a rational one. On the other hand, function T_i models the prescribed behavior (*proposed protocol*) for agent i, i.e. the behavior of *obedient* (or *altruistic*, following [1, 16]) agents. For any set Y of Byzantine and rational players such that all players not in Y are altruistic, the dynamic of the system is modeled by the transition relation $BT_{\mathcal{Q}}$ given in the following definition.

Definition 2. *Let* $\mathcal{Q} = \langle \boldsymbol{S}, \boldsymbol{I}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{B}, h, \beta \rangle$ *be an* n *players coalition schema. We define* $BT_{\mathcal{Q}} : \mathcal{P}([n]) \times \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S} \to \mathbb{B}$ *as follows:* $BT_{\mathcal{Q}}(Y, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'}) = \bigwedge_{i=1}^{n} BT_i(Y, \boldsymbol{s}, a_i, s'_i)$*, where*

$$BT_i(Y, \boldsymbol{s}, a_i, s'_i) = \begin{cases} B_i(\boldsymbol{s}, a_i, s'_i) & \text{if } i \in Y \\ B_i(\boldsymbol{s}, a_i, s'_i) \wedge T_i(\boldsymbol{s}, a_i) & \text{otherwise.} \end{cases}$$

We write BT *for* $BT_{\mathcal{Q}}$ *when* $\mathcal{Q}$ *is understood from the context.*

2.3 Mechanism

Coalitions are subsets of players that together act as a single rational player. This means that all players in a coalition aim at maximizing the *coalition reward*. This leads to the definition of *mechanism* (with coalitions).

Definition 3 (Mechanism). *A mechanism* $\mathcal{M}$ *is a pair* $\langle \mathcal{Q}, P \rangle$*, where* $\mathcal{Q}$ *is an* n *players coalition schema, and* $P = \{Q_1, \ldots, Q_m\}$ *is a partition of* $[n]$ *(thus each* Q_i *is a coalition).*

Remark 1. We can recover the definition of mechanism in [20] as a particular case of Def. 3, when all coalitions are singletons, i.e. $P = \{\{1\}, \ldots, \{n\}\}$, $h_i(\boldsymbol{s}, \boldsymbol{a}) = h(\{i\}, \boldsymbol{s}, \boldsymbol{a})$ is the reward of player i and $\beta_i = \beta(\{i\})$ is the discount factor of player i.

Remark 2. Given an n players mechanism $\mathcal{M} = \langle \mathcal{Q}, P \rangle$, where P is $\{Q_1, \ldots, Q_m\}$, there exists an equivalent m players mechanism $\hat{\mathcal{M}}$ with only singleton coalitions. The mechanism $\hat{\mathcal{M}}$ is defined as follows: $\hat{\mathcal{M}} = \langle \hat{\mathcal{Q}}, \hat{P} \rangle$, where $\hat{P} = \{\{1\}, \ldots, \{m\}\}$ and the coalition schema $\hat{\mathcal{Q}} = \langle \hat{\boldsymbol{S}}, \hat{\boldsymbol{I}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{T}}, \hat{\boldsymbol{B}}, \hat{h}, \hat{\beta} \rangle$ is defined as follows. The set of players of $\hat{\mathcal{Q}}$ is $[m]$. The set of states is $\hat{\boldsymbol{S}} = \langle \hat{S}_1, \ldots, \hat{S}_m \rangle$, where $\hat{S}_i = \prod_{j \in Q_i} S_j$. The set of initial states is $\hat{\boldsymbol{I}} = \langle \hat{I}_1, \ldots, \hat{I}_m \rangle$, where $\hat{I}_i = \prod_{j \in Q_i} I_j$. The set of actions is $\hat{\boldsymbol{A}} = \langle \hat{A}_1, \ldots, \hat{A}_m \rangle$, where $\hat{A}_i = \prod_{j \in Q_i} A_j$. If $\boldsymbol{s} = \langle s_1, \ldots, s_n \rangle \in \boldsymbol{S}$ then $\hat{\boldsymbol{s}} = \langle \boldsymbol{s}_{Q_1}, \ldots, \boldsymbol{s}_{Q_m} \rangle \in \hat{\boldsymbol{S}}$. If $\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle \in \boldsymbol{A}$ then $\hat{\boldsymbol{a}} = \langle \boldsymbol{a}_{Q_1}, \ldots, \boldsymbol{a}_{Q_m} \rangle \in \hat{\boldsymbol{A}}$. The underlying behavior of player i is $\hat{B}_i(\hat{\boldsymbol{s}}, \boldsymbol{a}_{Q_i}, \boldsymbol{s'}_{Q_i}) = \bigwedge_{j \in Q_i} B_j(\boldsymbol{s}, a_j, s_j)$. The proposed protocol for player i of $\hat{\mathcal{Q}}$ is $\hat{T}_i(\hat{\boldsymbol{s}}, \boldsymbol{a}_{Q_i}) = \bigwedge_{j \in Q_i} T_j(\boldsymbol{s}, a_j)$. The discount and reward functions for player i of $\hat{\mathcal{Q}}$ are: $\hat{\beta}(\{i\}) = \beta(Q_i)$ and $\hat{h}(\{i\}, \hat{\boldsymbol{s}}, \hat{\boldsymbol{a}}) = h(Q_i, \boldsymbol{s}, \boldsymbol{a})$.

Since our goal is checking that a given protocol is a Nash equilibrium with respect to *any coalition* of size at most q, we will be working, most of the time, using coalition schemas (Def. 1) rather than mechanisms (Def. 3). Finding a way (Sect. 5) to effectively represent coalition schemas is indeed one of our main contributions.

Without loss of generality, in what follows, we focus on mechanisms $\mathcal{M} = \langle \mathcal{Q}, P \rangle$, with at most one coalition of size greater than 1. That is, partitions P have the form $\{Q, \{j_1\}, \ldots, \{j_{m-1}\}\}$, with $|Q| \geq 1$. By abuse of notation, we denote such kind of partitions with the set of players $Q \subseteq [n]$ forming the non-singleton coalition.

Example 1 (Case Study 1). This case study presents a very simple scenario inspired by peer-to-peer streaming services. Ideally, $n \in \mathbb{N}$ agents have to collaborate to broadcast information in order to maintain a high-quality service. When an agent broadcasts information, it incurs a cost $c \in \mathbb{R}$. All agents have a profit $g \in \mathbb{R}$ if they collaborate and the number of collaborative agents exceeds a threshold pn, where $p \in (0, 1)$ is a real parameter. Otherwise the reward is 0. The set S_i of agent i local states is $\{0, 1\}$. The underlying behavior B_i of each agent $i \in [n]$ is depicted in Fig. 1: each agent in state 0 may choose whether to broadcast information (action *broadcast* which corresponds to the *proposed protocol* T_i) or do nothing (action *sleep*).

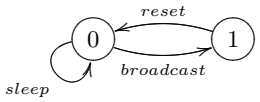

Fig. 1. Underlying behavior B_i for agent i.

Following the intuition that members of a coalition may broadcast information only to each other, the reward for a coalition Q is $g|Q|$ if at least pn agents (out of n) and at least $r|Q|$ ($r \in (0, 1)$) agents in the coalition have performed action *broadcast*. This last condition models the fact that a large enough number of agents must collaborate inside the coalition to maintain a high-quality service.

Codifying action *broadcast* with 1 and *sleep* with 0, we give the following formal definition of the reward function h (in state 1, 0 and 1 both codify action *reset*). Defining $f(Q, \boldsymbol{s}, \boldsymbol{a})$ as:

$$f(Q, \boldsymbol{s}, \boldsymbol{a}) = \begin{cases} g|Q| & \text{if } (\sum_{i \in [n]} s_i \geq pn) \wedge (\sum_{i \in Q} s_i \geq r|Q|) \\ 0 & \text{otherwise} \end{cases}$$

the reward function h is $h(Q, \boldsymbol{s}, \boldsymbol{a}) = f(Q, \boldsymbol{s}, \boldsymbol{a}) - c \sum_{i \in Q} \bar{s}_i a_i$.

3 Coalition Schemas as Games

A coalition schema $\mathcal{Q}$ induces a game that has all feasible paths as possible outcomes. The set of feasible paths depends on the $BT_{\mathcal{Q}}$ transition relation given in Def. 2. As a consequence, it depends on a set of players Y that may behave accordingly to the underlying behavior, whereas agents not in Y follow the proposed protocol. The set Y models both Byzantine and rational agents.

In the rest of this section, we assume an n players coalition schema $\mathcal{Q} = \langle \boldsymbol{S}, \boldsymbol{I}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{B}, h, \beta \rangle$ and a set of agents $Y \subseteq [n]$ to be given.

Paths. A *path* in $(\mathcal{Q}, Y)$ (or simply a *path* when $(\mathcal{Q}, Y)$ is understood from the context) is a (finite or infinite) sequence $\pi = \boldsymbol{s}(0)\boldsymbol{a}(0)\boldsymbol{s}(1) \ldots \boldsymbol{s}(t)\boldsymbol{a}(t)\boldsymbol{s}(t+1) \ldots$ where, for each t, $\boldsymbol{s}(t)$ is a global state, $\boldsymbol{a}(t)$ is a global action and $BT(Y, \boldsymbol{s}(t), \boldsymbol{a}(t), \boldsymbol{s}(t+1))$ holds. The *length* $|\pi|$ of a path π is the number of global actions in π. If π is infinite we write $|\pi| = \infty$.

In order to extract the t-th global state and the t-th global action from a given path π, we define $\pi^{(s)}(t) = \boldsymbol{s}(t)$ and $\pi^{(a)}(t) = \boldsymbol{a}(t)$. To extract actions of a set of players Q of size q, we denote with $\pi^{(a)}_Q(t)$ the q-tuple of actions $a_Q(t)$

at stage t of the set of agents Q and with $\pi^{(a)}_{-Q}(t)$ the actions $\boldsymbol{a}_{-Q}(t)$ at stage t of all agents not in Q.

For each set of agents $Q \subseteq [n]$, the *value* of a path π is $v(Q,\pi) = \sum_{t=0}^{|\pi|-1} \beta(Q)^t h\left(Q, \pi^{(s)}(t), \pi^{(a)}(t)\right)$. Note that for any path π and set of agents $Q \subseteq [n]$ the *path value* $v(Q,\pi)$ is well defined also when $|\pi| = \infty$ since the series $\sum_{t=0}^{\infty} \beta(Q)^t h\left(Q, \pi^{(s)}(t), \pi^{(a)}(t)\right)$ converges for all $\beta(Q) \in (0,1)$.

Given a path π and a non-negative integer $k \leq |\pi|$ we denote with $\pi|_k$ the *prefix* of π of length k, i.e. the finite path $\pi|_k = \boldsymbol{s}(0)\boldsymbol{a}(0)\boldsymbol{s}(1)\ldots\boldsymbol{a}(k-1)\boldsymbol{s}(k)$ and with $\pi|^k$ the *tail* of π, i.e. the path $\pi|^k = \boldsymbol{s}(k)\boldsymbol{a}(k)\boldsymbol{s}(k+1)\ldots\boldsymbol{s}(t)\boldsymbol{a}(t)\boldsymbol{s}(t+1)\ldots$.

We denote with $\mathrm{Path}_k(\boldsymbol{s}, Y)$ the set of all feasible paths of length k starting at $\boldsymbol{s}$, when Y is the set of rational/Byzantine agents. Formally, $\mathrm{Path}_k(\boldsymbol{s}, Y) = \{\pi \mid \pi \text{ is a path in } (\mathcal{Q}, Y) \text{ and } |\pi| = k \text{ and } \pi^{(s)}(0) = \boldsymbol{s}\}$. Since we aim to verify that a given protocol is robust with respect to all coalitions of size at most q and all sets of Byzantine agents of size at most f, we introduce the notation $\mathrm{Path}_k(\boldsymbol{s}, f, q)$ to denote the set of all paths of length k feasible with respect to all sets of Byzantine agents of cardinality at most f and all coalitions of size at most q. Formally, $\mathrm{Path}_k(\boldsymbol{s}, f, q) = \bigcup_{|Z|\leq f, 0<|Q|\leq q, Z\cap Q=\varnothing} \mathrm{Path}_k(\boldsymbol{s}, Z\cup Q)$. Unless otherwise stated, in the following we omit the subscript or superscript horizon when it is ∞. For example we write $\mathrm{Path}(\boldsymbol{s}, Y)$ for $\mathrm{Path}_\infty(\boldsymbol{s}, Y)$.

Let $W \subseteq [n]$ be a set of agents. A *path* π in $(\mathcal{Q}, Y)$ is said to be *W-altruistic* if for all $t < |\pi|$, and for all $i \in W$, $T_i(\pi^{(s)}(t), \pi^{(a)}_i(t))$ holds. Note that if $W \cap Y = \varnothing$, all paths in $\mathrm{Path}_k(\boldsymbol{s}, Y)$ are W-altruistic, that is, agents in W behave accordingly to the proposed protocol.

Strategies. As usual in a game theoretical setting, we need to distinguish a player actions (i.e. local actions) from those of its opponents. More in general, we need to distinguish actions of players in a coalition Q from those of players not in Q. This leads to the notion of strategy.

A *strategy* σ for a coalition Q is a (finite or infinite) sequence of actions tuples for the set of players Q. The *length* $|\sigma|$ of σ is the number of actions tuples in σ (thus if $|\sigma| = 0$, the strategy is empty). Let $\sigma = \boldsymbol{a}_0 \ldots \boldsymbol{a}_t \ldots$ be a strategy for coalition Q. We denote with $\sigma(t)$ the t-th action in σ, that is $\boldsymbol{a}_t$. Strategy σ *agrees* with a path π (notation $\pi \simeq^Q \sigma$) if $|\sigma| = |\pi|$ and for all $t < |\sigma|$, $\sigma(t) = \pi^{(a)}_Q(t)$. Given a path π, the strategy (of length $|\pi|$) for a coalition Q associated to π will be denoted by $\sigma(\pi, Q) = \pi^{(a)}_Q(0)\pi^{(a)}_Q(1)\ldots\pi^{(a)}_Q(t)\ldots$.

For any set of agents $Y \subseteq [n]$ the set of *Y-feasible strategies* of length k for a coalition Q in state $\boldsymbol{s}$ is: $\mathrm{Strat}_k(\boldsymbol{s}, Q, Y) = \{\sigma(\pi, Q) \mid \pi \in \mathrm{Path}_k(\boldsymbol{s}, Y)\}$. Our definition of feasible strategy essentially corresponds to the usual one in multi-stage games: global states implicitly represent the sequence of actions in previous periods, i.e. histories. In contrast with the game theoretical model, histories are partitioned into a finite number of equivalence classes, represented as mechanism states.

As for paths, a strategy $\sigma \in \mathrm{Strat}_k(\boldsymbol{s}, Q, Y)$ is said to be *Q-altruistic* if $Q \cap Y = \varnothing$. If $\sigma = a_0 a_1 \ldots a_{k-1} a_k a_{k+1} \ldots$, we use the notations $\sigma|_k = a_0 a_1 \ldots a_{k-1}$ and $\sigma|^k = a_k a_{k+1} \ldots$ to denote, respectively, the k-prefix and the tail of σ. The

set of paths that agree with a set of strategies Σ for a coalition Q is defined as $\mathrm{Path}(\boldsymbol{s}, Q, Y, \Sigma) = \{\pi \in \mathrm{Path}_k(\boldsymbol{s}, Y) \mid \exists \sigma \in \Sigma.\ k = |\sigma| \wedge \pi \simeq^Q \sigma\}$. When Σ is the singleton $\{\sigma\}$, we simply write $\mathrm{Path}(\boldsymbol{s}, Q, Y, \sigma)$.

Given a set of Byzantine agents $Z \subseteq [n] \setminus Q$, the *guaranteed outcome* (or the *value*) of a strategy σ in state $\boldsymbol{s}$ for coalition Q is the minimum value of paths that agree with σ. Formally: $v(\boldsymbol{s}, Q, Z, \sigma) = \min\{v(Q, \pi) \mid \pi \in \mathrm{Path}(\boldsymbol{s}, Q, Z \cup Q, \sigma)\}$. The *value* of a state $\boldsymbol{s}$ at horizon k for coalition Q is the guaranteed outcome of the best strategy of length k starting at state $\boldsymbol{s}$. Formally: $v^k(\boldsymbol{s}, Q, Z) = \max\{v(\boldsymbol{s}, Q, Z, \sigma) \mid \sigma \in \mathrm{Strat}_k(\boldsymbol{s}, Q, Z \cup Q)\}$. The *guaranteed outcome of the proposed protocol* in a state $\boldsymbol{s}$ at horizon k for coalition Q is the outcome of the worst Q-altruistic strategy of length k starting at state $\boldsymbol{s}$. Formally, $u^k(\boldsymbol{s}, Q, Z) = \min\{v(\boldsymbol{s}, Q, Z, \sigma) \mid \sigma \in \mathrm{Strat}_k(\boldsymbol{s}, Q, Z)\}$.

The finite horizon value of a state can be effectively computed by using a dynamic programming approach (Prop. 1). This is one of the main ingredients of our verification algorithm (Sect. 5). We omit proofs because of lack of space.

Proposition 1. *Let $\mathcal{Q} = \langle \boldsymbol{S}, \boldsymbol{I}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{B}, h, \beta \rangle$ be an n players coalition schema, $\boldsymbol{s} \in S$, and $Q, Z \subseteq [n]$ such that $Z \cap Q = \varnothing$. The state values at horizon k for coalition Q can be computed as follows:*

$$v^0(\boldsymbol{s}, Q, Z) = u^0(\boldsymbol{s}, Q, Z) = 0;$$

$$v^{k+1}(\boldsymbol{s}, Q, Z) = \max_{\boldsymbol{a}_Q \in A_Q} \min_{\boldsymbol{a_{-Q}} \in A_{-Q}} \{\, h(Q, \boldsymbol{s}, \langle \boldsymbol{a}_Q, \boldsymbol{a}_{-Q} \rangle) + \beta(Q)\, v^k(\boldsymbol{s'}, Q, Z) \mid BT(Z \cup Q, \boldsymbol{s}, \langle \boldsymbol{a}_Q, \boldsymbol{a}_{-Q} \rangle, \boldsymbol{s'})\};$$

$$u^{k+1}(\boldsymbol{s}, Q, Z) = \min_{\boldsymbol{a}_Q \in A_Q} \min_{\boldsymbol{a}_{-Q} \in A_{-Q}} \{\, h(Q, \boldsymbol{s}, \langle \boldsymbol{a}_Q, \boldsymbol{a}_{-Q} \rangle) + \beta(Q)\, u^k(\boldsymbol{s'}, Q, Z) \mid BT(Z, \boldsymbol{s}, \langle \boldsymbol{a}_Q, \boldsymbol{a}_{-Q} \rangle, \boldsymbol{s'})\}.$$

Example 2. In the Coalition Schema described in Ex. 1, a rational player may deviate from the proposed protocol if it thinks that the service is compromised because the number of Byzantine agents is larger than $(1-p)n$. In such a case a rational player choose the action *sleep*, which ensures a reward 0, rather than the action *broadcast*, which leads to the negative reward $c \sum_{k \in \mathbb{N}} \beta^{2k}$. A coalition Q may deviate by using the following strategy: some agents broadcast information to other coalition members only, and some agents do not broadcast anything. In such a case, threshold $r|Q|$ of collaborative agents is required to guarantee the service for coalition members. As a consequence, the coalition Q deviates whenever $\lceil r|Q| \rceil < |Q|$. As expected, if all agents are in the coalition, the protocol is not Nash, but the reward of each agent increases (the well known *price of anarchy* phenomenon[15]).

4 Verifying Coalition Nash Equilibria

In this section, we introduce the notion of ε-f-q-Nash coalition schema, in order to verify protocol robustness with respect to coalitions of colluding players. Theor. 1 gives sufficient conditions to check that a coalition schema is ε-f-q-Nash and, together with Prop. 1, it proves the correctness of our verification (Alg. 1 in Sect. 5).

In [20] it is introduced a notion of ε-f-Nash equilibrium for a mechanism. Intuitively, a mechanism $\mathcal{M}$ is ε-f-Nash, if, as long as the number of Byzantine agents is no more than f (e.g. see [11]), no rational agent has an interest greater than ε (e.g. see [12, 14]) in deviating from the proposed protocol in $\mathcal{M}$. ε-f-q-Nash extends this notion by requiring that no coalition of rational agents of size at most q has an interest greater than ε in deviating from the proposed protocol.

Definition 4 (ε-f-q-Nash). *Let $\mathcal{Q} = \langle \boldsymbol{S}, \boldsymbol{I}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{B}, h, \beta \rangle$ be an n players coalition schema. Let $0 \neq q \in [n]$, $f \leq n - q$, $\varepsilon > 0$ and $Q \subseteq [n]$ be a coalition with $0 < |Q| \leq q$.*

The coalition schema $\mathcal{Q}$ is ε-f-Nash for coalition Q iff $\forall Z \in \mathcal{P}_f([n] \setminus Q)$, $\forall \boldsymbol{s} \in I$, we have:

$$u(\boldsymbol{s}, Q, Z) + \varepsilon \geq v(\boldsymbol{s}, Q, Z).$$

$\mathcal{Q}$ is ε-f-q-Nash if it is ε-f-Nash for all coalitions Q such that $0 < |Q| \leq q$.

In general, Nash equilibria for infinite-horizon games cannot be verified by only looking at finite strategies, since they are not necessarily limits of equilibria of the corresponding finite horizon games (e.g. see [14], or Ex. 1 in [20] for an example in mechanism scenario). However, if we assume that agents cannot distinguish between small variations (ε) in their payoffs, then we can verify Nash equilibria for infinite-horizon games by only looking at *long enough* finite strategies. This has motivated the introduction of ε-Nash equilibria for infinite-horizon games and also motivates Def. 4. Indeed, our definition of ε-0-1-Nash yields the usual definition of ε-Nash equilibria (e.g., see Sect. 4.8 of [14]). We observe that the notion of ε-f-1-Nash is equivalent to the notion of ε-f-Nash in [20]. Thus if a mechanism is ε-f-q-Nash for $q \geq 1$, it is also ε-f-Nash.

ε-f-q-Nash property cannot be verified using finite approximations if for some $Q, Z, \boldsymbol{s}$, $| v^k(\boldsymbol{s}, Q, Z) - u^k(\boldsymbol{s}, Q, Z) |$ converges to ε (see Ex. 2 in [20]). However we may get arbitrarily close to this result as stated by the following theorem.

Theorem 1. *Let $\mathcal{Q} = \langle \boldsymbol{S}, \boldsymbol{I}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{B}, h, \beta \rangle$ be an n players coalition schema. Let $0 \neq q \in [n]$, $f \leq n - q$, $\varepsilon > 0$ and $\delta > 0$. Furthermore, for each coalition Q, such that $0 < |Q| \leq q$ let:*

1. $M_Q = \max\{|h(Q, \boldsymbol{s}, \boldsymbol{a})| \mid \boldsymbol{s} \in S' \text{ and } \boldsymbol{a} \in A'\}$.
2. $E(Q, k) = 5\, \beta(Q)^k \frac{M_Q}{1 - \beta(Q)}$.
3. $\Delta(Q, k) = \max\{v^k(\boldsymbol{s}, Q, Z) - u^k(\boldsymbol{s}, Q, Z) \mid \boldsymbol{s} \in I, Z \in \mathcal{P}_f([n] \setminus Q)\}$.
4. $\varepsilon_1(Q, k) = \Delta(Q, k) - 2E(Q, k)$.
5. $\varepsilon_2(Q, k) = \Delta(Q, k) + 2E(Q, k)$.

Let k be such that for all coalitions Q such that $0 < |Q| \leq q$, $4\,E(Q, k) < \delta$ holds. Then we have:

1. *If for all $Q \in \mathcal{P}_q([n])$ $\varepsilon \geq \varepsilon_2(Q, k) > 0$ then $\mathcal{Q}$ is ε-f-q-Nash.*
2. *If there is $Q \in \mathcal{P}_q([n])$ such that $0 < \varepsilon \leq \varepsilon_1(Q, k)$ then $\mathcal{Q}$ is not ε-f-q-Nash. Of course in such a case a fortiori $\mathcal{Q}$ is not 0-f-q-Nash.*
3. *If for all $Q \in \mathcal{P}_q([n])$, $\varepsilon_1(Q, k) < \varepsilon$ and there exists $Q' \in \mathcal{P}_q([n])$ s.t. $\varepsilon < \varepsilon_2(Q', k)$ then $\mathcal{Q}$ is $(\varepsilon + \delta)$-f-q-Nash.*

Proof. The proof skeleton is essentially the following: first, we show that values of path prefixes converge to the path values, i.e. $v(Q, \pi\,|_k) \rightarrow v(Q, \pi)$. Then, we show that values of strategy prefixes converge to the strategy values, i.e. $v(\boldsymbol{s}, Q, Z, \sigma|_k) \rightarrow v(\boldsymbol{s}, Q, Z, \sigma)$, and finally we show that state values are limits of their finite approximations, i.e. $v^k(\boldsymbol{s}, Q, Z) \rightarrow v(\boldsymbol{s}, Q, Z)$. Bounds in convergence proofs give us the effective test to check if a mechanism is ε-f-q-Nash.

5 Verification Algorithms

Resting on Theor. 1 in this section we present our algorithm (Alg. 1) to verify that a given protocol is ε-f-q-Nash.

Alg. 1 is implemented on top of the NuSMV [22] model checker. Since states ($\boldsymbol{s}$), actions ($\boldsymbol{a}$) and sets of agents (Q, Z) are finite, we can represent them with boolean arrays. We represent boolean functions (such as the transition relations T, B and $BT_{\mathcal{Q}}$) using OBDDs [3] and real valued functions (such as coalition rewards $\lambda \boldsymbol{s}\ Q\ Z.\ u^t(\boldsymbol{s}, Q, Z)$ and $\lambda \boldsymbol{s}\ Q\ Z.\ v^t(\boldsymbol{s}, Q, Z)$) using ADDs (*Arithmetic Decision Diagrams*) as implemented in the CUDD [10] package.

As usual in OBDD based computations, we represent functions with the expressions defining them. For the sake of clarity, we will present Alg. 1 using a set theoretic notation for sets, predicates and functions over sets, but for example statements in lines 2, 5, 7, and 11 have to be interpreted as ADD manipulations.

The algorithm first computes the number of iterations k needed to reach the precision threshold desired by the user (line 1). Then, for each iteration t from 0 to k, it computes the state value $v^t(\boldsymbol{s}, Q, Z)$ and the proposed protocol guaranteed outcome $u^t(\boldsymbol{s}, Q, Z)$ using Prop. 1 (lines 2–10). After computing state values, Alg. 1 finds the ADD representing the maximum difference between state values and the guaranteed outcome of the proposed protocol as a function of Q. This is the $\Delta(Q)$ in line 11, representing the best gain that a coalition Q may have in deviating from the proposed protocol. This corresponds to point 3 of Theor. 1. Then in line 12, $\varepsilon_1(Q)$ and $\varepsilon_2(Q)$ are computed as in points 4 and 5 of Theor. 1 respectively. Finally, lines 13–15 determine which statement among 1–3 of Theor. 1 holds.

Verifying that a coalition schema $\mathcal{Q}$ is ε-f-q-Nash requires checking that the hypotheses in statements 1-3 of Theor. 1 hold for all coalitions of size at most q. The number of such coalitions is $\sum_{j=1}^{q} \binom{n}{j}$. We implement two versions of Alg. 1

- in the *explicit* version, the loop in lines 3-10 is performed $\sum_{j=1}^{q} \binom{n}{j}$ times in order to compute state values $u^t(\boldsymbol{s}, Q, Z)$ and $v^t(\boldsymbol{s}, Q, Z)$ for any possible coalition Q of size at most q. This is almost equivalent to build the single player mechanism $\mathcal{M}_Q$ for each coalition Q and to run (a variation of) the single player verification algorithm presented in [20] with $\mathcal{M}_Q$ as input;
- in the *symbolic* version, all computations are parametric with respect to all mechanisms $\mathcal{M}_Q$: in such a case, line 3 has to be read as a logical predicate rather that an iterative **for** loop.

Algorithm 1 *CheckNash*. Checking if a mechanism is ε-f-q-Nash

Input: mechanism $\mathcal{Q}$, int f, int q, double ε, δ
Output: (FAIL) or (PASS with a threshold)
1: Let k be such that $\forall Q\ 0 < |Q| \leq q \Rightarrow [4\,E(Q,k) < \delta]$
2: $v^0(\boldsymbol{s}, Q, Z) \leftarrow 0$, $u^0(\boldsymbol{s}, Q, Z) \leftarrow 0$, where $\boldsymbol{s} \in \boldsymbol{S}, Q \in \mathcal{P}_q([n])$ and $Z \in \mathcal{P}_f([n] \setminus Q)$
3: **for all** $Q \in \mathcal{P}_q([n]) \setminus \{\varnothing\}$ **do**
4: **for** $t = 1$ **to** k **do**
5: $v^t(\boldsymbol{s}, Q, Z) \leftarrow \max_{a_Q \in A_Q} \min_{\boldsymbol{a}_{-Q} \in A_{-Q}} [h_i(Q, \boldsymbol{s}, \langle a_Q, \boldsymbol{a}_{-Q}\rangle) + \beta(Q) v^{t-1}(\boldsymbol{s'}, Q, Z)]$,
6: where $BT(Q \cup Z, \boldsymbol{s}, \langle a_Q, \boldsymbol{a}_{-Q}\rangle, \boldsymbol{s'}), \boldsymbol{s} \in \boldsymbol{S}, Q \in \mathcal{P}_q([n])$ and $Z \in \mathcal{P}_f([n] \setminus Q)$
7: $u^t(\boldsymbol{s}, Q, Z) \leftarrow \min_{a_Q \in A_Q} \min_{\boldsymbol{a}_{-Q} \in A_{-Q}} [h_i(Q, \boldsymbol{s}, \langle a_Q, \boldsymbol{a}_{-Q}\rangle) + \beta(Q) u^{t-1}(\boldsymbol{s'}, Q, Z)]$,
8: where $BT(Z, \boldsymbol{s}, \langle a_i, \boldsymbol{a}_{-i}\rangle, \boldsymbol{s'}), \boldsymbol{s} \in \boldsymbol{S}, Q \in \mathcal{P}_q([n])$, and $Z \in \mathcal{P}_f([n] \setminus Q)$
9: **end for**
10: **end for**
11: $\Delta(Q) \leftarrow \max\{v^k(\boldsymbol{s}, Q, Z) - u^k(\boldsymbol{s}, Q, Z) \mid \boldsymbol{s} \in I, Z \in \mathcal{P}_f([n] \setminus Q)\}$, $Q \in \mathcal{P}_q([n])$
12: $\varepsilon_1(Q) \leftarrow \Delta(Q) - 2E(Q)$, $\varepsilon_2(Q) \leftarrow \Delta(Q) + 2E(Q)$, with $Q \in \mathcal{P}_q([n])$
13: **if** $(\exists Q \in \mathcal{P}_q[n]\ [\varepsilon < \varepsilon_1(Q)])$ **return** (FAIL)
14: **if** $(\forall Q \in \mathcal{P}_q[n]\ [\varepsilon_2(Q) < \varepsilon])$ **return** (PASS with ε)
15: **else return** (PASS with $(\varepsilon + \delta)$)

Symbolic approach should be asymptotically better. However it requires the introduction of auxiliary state and action variables for maximin computations required by Alg. 1 in lines 5 and 7, which turns out to be much more involved and slower. Moreover, ADDs make usual OBDD memory compression via sharing much less effective. As experimental results show in Sect. 6, in our moderate size mechanisms the explicit implementation outperforms the symbolic one, both in running time and in memory usage.

6 Experimental Results

In order to assess effectiveness of our Nash verifier we present experimental results on its usage on two meaningful and scalable case studies inspired by cooperative services. Case study 1 shown in Ex. 1 is designed to be as simplest as possible, in order to test our verification tool on mechanism with a number of agents as greater as possible. In Sect. 6.1 case study 2 is presented, describing a slightly more complex scenario. Finally, Sect. 6.2 describes experimental settings and assesses tool performances.

6.1 Case Study 2

In this case study we present a slightly more complex and subtle scenario. We are given a set $\mathcal{J} = \{0, \ldots, m-1\}$ of m *jobs* and a set $\mathcal{T} = \{0, \ldots, t-1\}$ of t *tasks*. Function $\eta : \mathcal{J} \to \mathcal{P}(\mathcal{T})$ defines for each job j the set of tasks $\eta(j)$ needed to complete j, and function $\iota : \mathcal{T} \to \mathcal{P}(\mathcal{J})$ defines for each task t the set of jobs $\iota(t)$, for which t is needed.

Each agent $i \in [n]$ is supposed to work (*proposed protocol*) on a given sequence of (not necessarily distinct) tasks $\mathcal{T}_i = \langle \tau(i,0), \ldots, \tau(i,\alpha(i)-1) \rangle$ starting from $\tau(i,0)$ and returning to $\tau(i,0)$ after task $\tau(i,\alpha(i)-1)$ has been completed. An agent may *deviate* from the proposed protocol by not executing the task. This behavior models many typical scenarios in cooperative services.

An agent incurs a *cost* w by working towards the completion of its currently assigned task. A job is completed if for each task it needs there exists at least one agent that has completed that task. In such a case, each of such agents receives a *reward*. At the end of each *round* a fixed capital C is equally divided among completed jobs. The reward for a job is in turn equally divided among all agents that executed a task needed to complete the job. Note that even if two (or more) agents have completed the same task all of them get a reward. Finally, we set the reward for a coalition Q to be the sum of its component rewards.

Note that, in this scenario, a coalition may deviate from the proposed protocol in the following way: if two or more players in the coalition are assigned to the same task, it is sufficient that only one of them works. Moreover, if there is a large enough number of Byzantine players, jobs completion is not guaranteed. This may induce a rational player not to work, in order to avoid the cost w.

6.2 Results

In this section we summarize the experimental results we obtain by running the Nash verification algorithm Alg. 1 in both its implementations (i.e. explicit and symbolic) on our case studies.

Results on Case Study 1. We instantiate the class of mechanisms related to our first case study by fixing $p = \frac{3}{4}$, $g = 4$ and $r = \frac{1}{2}$. We then perform our experiments both with the explicit and the symbolic implementation of Alg. 1, by increasing the number of agents n, the coalition maximum size q and the Byzantine agents maximum number b. We set for all coalitions Q, $\beta(Q) = 0.5$.

Results are in Tab. 1. Column meanings in Tab. 1 are as follows. Columns n, q, b show the number of agents, the maximum coalition size, and the maximum number of Byzantine agents. Column **Nash** shows the final verification outcome. In particular, note that the PASS result, in our experiments, always means PASS with ε (case 1 of Theor. 1). Column **CPU expl** (resp., **symb**) shows the computation time in seconds for the explicit (resp. symbolic) implementation. Column **Mem expl** (resp., **symb**) shows the RAM used by the explicit (resp. symbolic) implementation in MBs. Column **BDD expl** (resp., **symb**) shows the number of OBDD/ADD nodes used by the explicit (resp. symbolic) implementation. Finally, column $|\mathcal{P}_q([n])|$ shows the number iterations needed by line 3 of Alg. 1, i.e. $|\mathcal{P}_q([n]) \setminus \{\varnothing\}|$.

Note that "N/A" entries denotes that the corresponding experiment exceeded the available resources, either w.r.t. RAM (needed more than 8 GB) or time (needed more than 3 days). In all experiments we take $\varepsilon = 0.1$ and accuracy $\delta = 0.05$. With such settings the value k in line 1 of Alg. 1 turns out to be at most 15 in all our experiments.

Table 1. Experiments run for the case study of Sect. 1 on a 64-bit Dual Quad Core 3 GHz Intel Xeon Linux PC with 8 GB of RAM

n	q	b	Nash	CPU expl	CPU symb	Mem expl	Mem symb	BDD expl	BDD symb	$\lvert\mathcal{P}_q([n])\rvert$
9	1	3	PASS	2.88e+01	2.82e+03	7.43e+01	1.63e+02	2.59e+05	8.25e+05	9
9	1	4	FAIL	3.03e+01	5.87e+03	7.41e+01	1.63e+02	5.36e+05	7.77e+05	9
9	2	3	PASS	9.26e+01	1.79e+04	7.65e+01	1.64e+02	5.66e+05	1.16e+06	45
9	2	4	FAIL	1.33e+02	3.01e+04	7.54e+01	1.66e+02	2.92e+05	1.67e+06	45
9	3	0	FAIL	1.31e+01	7.73e+02	6.97e+01	1.62e+02	3.91e+05	5.31e+05	129
9	4	0	FAIL	2.25e+01	2.22e+03	7.27e+01	1.63e+02	4.54e+05	3.87e+05	255
9	5	0	FAIL	4.14e+01	6.20e+03	7.61e+01	1.63e+02	5.21e+05	9.30e+05	381
9	6	0	FAIL	7.90e+01	1.03e+04	7.68e+01	1.64e+02	5.95e+05	1.18e+06	465
9	7	0	FAIL	9.01e+01	1.52e+04	7.69e+01	1.63e+02	7.97e+05	1.25e+06	501
9	8	0	FAIL	8.24e+01	1.52e+04	7.69e+01	1.64e+02	7.39e+05	1.45e+06	510
9	9	0	FAIL	8.42e+01	1.58e+04	7.69e+01	1.64e+02	4.55e+05	1.45e+06	511
10	2	3	PASS	2.16e+02	9.21e+04	7.82e+01	5.41e+02	5.79e+05	2.63e+06	55
10	2	4	FAIL	2.70e+02	1.61e+05	8.05e+01	5.45e+02	5.54e+05	3.66e+06	55
11	2	3	PASS	5.97e+02	N/A	8.47e+01	N/A	9.90e+05	2.32e+06	66
16	2	5	FAIL	1.07e+05	N/A	4.48e+03	N/A	7.97e+07	N/A	136
18	2	4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	171

Results on Case Study 2. We instantiate the class of mechanisms related to our second case study as follows. First of all, we take the number n of agents to be greater than or equal to the number t of tasks. Second, we take the number m of jobs to be equal to t. Third, we define $\eta(j)$ (i.e. the set of tasks needed to complete job j) as follows: $\eta(j) = \{j, (j+1) \bmod t\}$. That is, each job requires two tasks and each task participates in two jobs. We take as task sequence for agent i the sequence $\mathcal{T}_i = \langle (i-1) \bmod t, \ldots, t-1, 0, 1, \ldots, ((i-1) \bmod t)-1 \rangle$. In other words, all agents consider tasks with the same order (namely $\langle 0, \ldots, t-1 \rangle$). The only difference is that agent i will start its task sequence from task $(i-1) \bmod t$. We set for all coalitions Q, $\beta(Q) = 0.5$. Finally, in order to ease computations we set the cost of working on a task $w = 1440$ and the capital to be divided among completed jobs $C = 4320n$. With the above settings we have the following parameters to be instantiated: n (number of agents), m (number of jobs), q (number of agents in a coalition).

Tab. 2 shows our experimental results on verification of the ε-f-q-Nash property for our case study. Column meanings in Tab. 2 are the same of Tab. 1, with the only addition of column m showing the number of jobs.

In all experiments we take $\varepsilon = 0.1$ and accuracy $\delta = 0.01$. With such settings the value k in line 1 of Alg. 1 turns out to be at most 27 in all our experiments.

Experimental Results Summary. From Tab. 1 and 2 we see that we can effectively handle moderate size mechanisms. Such mechanisms correspond indeed to quite large games. In fact, given a finite horizon k, an n players mechanism can be seen as a game which outcomes are n-tuples $\langle \sigma_1, \ldots, \sigma_n \rangle$ of strategies of length k, where σ_i is the strategy played by agent i. If the underlying behavior

Table 2. Experiments run for the case study of Sect. 6.1 on a 64-bit Dual Quad Core 3 GHz Intel Xeon Linux PC with 8 GB of RAM

n	m	q	b	Nash	CPU expl	CPU symb	Mem expl	Mem symb	BDD expl	BDD symb	$\lvert\mathcal{P}_q([n])\rvert$
4	2	2	0	PASS	1.40e+00	2.74e+00	8.82e+01	8.70e+01	3.99e+05	9.05e+04	10
4	2	4	0	PASS	2.45e+00	4.78e+00	8.82e+01	8.68e+01	8.53e+04	1.14e+05	15
5	2	3	2	FAIL	3.92e+01	2.12e+02	1.14e+02	1.14e+02	2.49e+05	9.27e+05	25
5	2	5	0	PASS	1.36e+01	5.12e+01	1.07e+02	1.14e+02	5.74e+05	6.22e+05	31
6	2	3	3	FAIL	3.33e+02	5.71e+03	1.27e+02	1.87e+02	7.57e+05	1.95e+06	41
6	2	6	0	PASS	8.67e+01	7.61e+02	1.25e+02	1.29e+02	5.68e+05	8.99e+05	63
6	3	1	3	PASS	2.55e+03	4.15e+04	2.57e+02	6.50e+02	2.44e+06	8.19e+06	6
6	3	1	4	FAIL	3.43e+03	5.87e+04	2.71e+02	8.70e+02	2.69e+06	9.77e+06	6
6	3	2	3	PASS	8.96e+03	1.70e+05	2.62e+02	1.70e+03	2.88e+06	1.90e+07	21
6	3	2	4	FAIL	1.04e+04	2.01e+05	2.95e+02	1.75e+03	3.59e+06	4.56e+06	21
6	3	3	0	FAIL	1.23e+03	5.80e+03	1.79e+02	3.64e+02	1.85e+06	4.60e+06	41
6	3	4	0	FAIL	1.88e+03	1.11e+04	1.79e+02	5.11e+02	2.03e+06	6.66e+06	56
6	3	5	0	FAIL	2.20e+03	1.56e+04	1.81e+02	5.27e+02	2.12e+06	6.76e+06	62
6	3	6	0	FAIL	2.28e+03	1.70e+04	1.81e+02	5.34e+02	1.93e+06	9.50e+06	63

of agent i allows two actions for each state, then there are 2^k strategies available for agent i. This would yield a game which normal form has 2^{kn} entries. In the coalition schemas used in Tab. 1, each agent can choose at least among $fib(k)$ (the k-th Fibonacci number) strategies. With n players this yields a normal form game with $fib(k)^n$ entries. If we look at coalitions of size j we have to consider $\binom{n}{j}$ games of size $fib(k)^n$. Since we are considering coalitions of size up to q we are indeed looking at $\sum_{j=1}^{q}\binom{n}{j}$ games of size $fib(k)^n$. For example, with horizon $k = 20$ and $n = 6$ the rows in Tab. 2 with $q = 2$ entail checking Nash equilibria for 21 games each of size $fib(19)^6 = 4181^6 \approx 5 \times 10^{21}$.

For both our case studies, the explicit implementation (in the sense of Sect. 5) outperforms the symbolic one both in RAM and computation time. In particular note that in Tab. 1, for $n = 9, b = 0, q \in [3, 9]$ the number of coalition grows but the symbolic algorithm does not take any advantage of this.

7 Conclusions

We presented two algorithms based on symbolic model checking for verification of Nash equilibria in finite state mechanisms modeling MAD distributed systems with coalitions. The first algorithm, explicitly checks Nash equilibria for any possible coalition within a given size, while the second symbolically represents all coalitions. An experimental comparison shows that the explicit one performs better. Moreover, our experiments show effectiveness of the presented algorithms for moderate size mechanisms. For example, we can handle mechanisms which corresponding normal form games would have more than 5×10^{21} entries.

Future research work include: investigation of more efficient algorithms in order to handle larger size mechanisms, exploiting symmetries in the definition of the mechanism.

References

1. Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. Bar fault tolerance for cooperative services. In *Proc. of SOSP'05*, pages 45–58, 2005. ACM Press.
2. Christopher Batten, Kenneth Barr, Arvind Saraf, and Stanley Trepetin. pStore: A secure peer-to-peer backup system. Technical Memo MIT-LCS-TM-632, Massachusetts Institute of Technology Laboratory for Computer Science, October 2002.
3. R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. on Computers*, C-35(8):677–691, Aug 1986.
4. Levente Buttyán and Jean-Pierre Hubaux. *Security and Cooperation in Wireless Networks - Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing (version 1.5)*. Cambridge University Press, 2007.
5. Steve Chien and Alistair Sinclair. Convergence to approximate nash equilibria in congestion games. In *Proc. of SODA'07*, pages 169–178, 2007.
6. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.
7. A. Clement, H. Li, J. Napper, J-P Martin, L. Alvisi, and M. Dahlin. BAR primer. In *Proc. of DSN'08*, 2008.
8. Bram Cohen. Incentives build robustness in bittorrent. In *Proc. IPTPS'03*, 2003.
9. Landon P. Cox and Brian D. Noble. Samsara: honor among thieves in peer-to-peer storage. In *Proc. of SOSP'03*, pages 120–132, 2003. ACM.
10. CUDD Web Page: http://vlsi.colorado.edu/∼fabio/, 2004.
11. Kfir Eliaz. Fault tolerant implementation. *Review of Economic Studies*, 69(3):589–610, July 2002.
12. H. Everett. Recursive games. *Contributions to the theory of games, vol. III - Annals of Mathematical Studies*, 39, 1957.
13. Joan Feigenbaum, Rahul Sami, and Scott Shenker. Mechanism design for policy routing. In *Proc. of PODC '04*, pages 11–20, 2004. ACM.
14. D. Fudenberg and J. Tirole. *Game theory*. MIT Press, aug 1991.
15. Ara Hayrapetyan, Éva Tardos, and Tom Wexler. The effect of collusion in congestion games. In *Proc. of STOC '06*, pages 89–98, 2006. ACM.
16. H. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR gossip. In *Proc. of OSDI'06*, 2006.
17. Mark Lillibridge, Sameh Elnikety, Andrew Birrell, Mike Burrows, and Michael Isard. A cooperative internet backup scheme. In *Proc. of ATEC'03*, pages 3–3, 2003. USENIX Association.
18. Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Sustaining cooperation in multi-hop wireless networks. In *Proc. of NSDI'05*, pages 231–244, 2005. USENIX Association.
19. Petros Maniatis, David S. H. Rosenthal, Mema Roussopoulos, Mary Baker, TJ Giuli, and Yanto Muliadi. Preserving peer replicas by rate-limited sampled voting. In *Proc. SOSP'03*, pages 44–59, 2003. ACM.
20. F. Mari, I. Melatti, I. Salvo, E. Tronci, L. Alvisi, A. Clement, and H. Li. Model checking nash equilibria in mad distributed system. In A. Cimatti and R. B. Jones, editors, *Proc. of FMCAD'08*, pages 85–92. IEEE, 2008.
21. Noam Nisan and Amir Ronen. Algorithmic mechanism design (extended abstract). In *Proc. of STOC'99*, pages 129–140, 1999. ACM.
22. NuSMV Web Page: http://nusmv.irst.itc.it/, 2006.
23. Jeffrey Shneidman and David C. Parkes. Specification faithfulness in networks with rational nodes. In *Proc. of PODC'04*, pages 88–97, 2004. ACM.